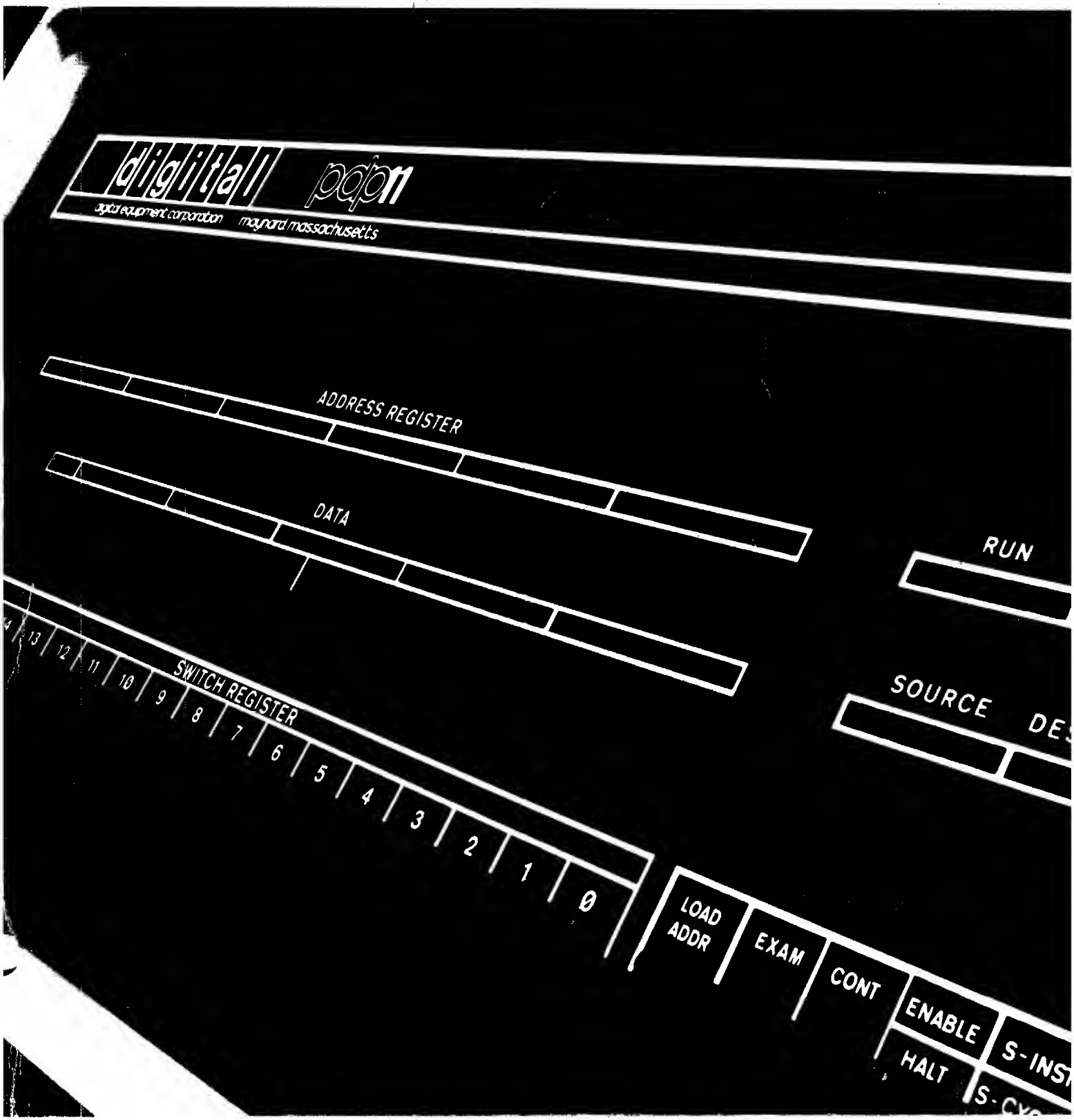
digital

pdp11

unibus interface manual



DEC-11-HIAA-D

PDP-11
INTERFACE MANUAL

DIGITAL EQUIPMENT CORPORATION □ MAYNARD, MASSACHUSETTS

APRIL 1970

TABLE OF CONTENTS

TABLE OF CONTENTS (cont)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (cont)

LIST OF TABLES

Chapter 1
General Description

1.1 SCOPE

This manual provides detailed interfacing information for PDP-11 System users. It is assumed that the reader is familiar with the PDP-11 System and has a thorough understanding of the principles and concepts introduced in the PDP-11 Handbook. This handbook presents a comprehensive overview of the system, a detailed description of the instruction set, and details of peripheral programming.

This manual describes detailed operation of the Unibus and methods for interfacing to the Unibus for the addition of custom-designed peripheral equipment to the PDP-11 System. It is beyond the scope of this manual to provide detailed theory of operation for the central processor or the instruction set, however, a short description of the processor is presented in Appendix A, and a description of bus cycles relating to the instruction set is presented in Appendix B.

1.2 APPLICABLE DOCUMENTS

The following documents will help the reader understand interface techniques and the overall PDP-11 System:

a. PDP-11 Handbook
b. KA11 Central Processor Manual
c. Digital 1970 Logic Handbook

The PDP-11 Handbook introduces the overall system and describes the instruction set. The PDP-11 Interface Manual and the KA11 Central Processor Manual, used together, completely describe the PDP-11 System. The primary subject of the PDP-11 Interface Manual is the Unibus; the primary subject of the KA11 Manual is the central processor. Many logic circuits used to implement the interface examples are covered in the Digital 1970 Logic Handbook.

1.3 INTRODUCTION

Digital Equipment Corporation's PDP-11 is a 16-bit, general-purpose, parallel-logic computer that uses 2's complement arithmetic. All communication between system components is accomplished by a single high-speed bus called the Unibus. Four concepts are extremely important for an understanding of the hardware and software implications of the Unibus. Each concept is covered separately in subsequent paragraphs.

1.3.1 Single Bus

The Unibus is a single, common path that connects the central processor, memory, and all peripherals. Addresses, data, and control information are transmitted along the 56 lines of the bus. Figure 1-1 is a simplified block diagram of the PDP-11 System and Unibus.

The form of communication is the same for every device on the Unibus. The processor uses the same set of signals to communicate with memory and peripheral devices. Peripheral devices also use this set of signals when communicating with the processor, memory, or other peripheral devices.

All instructions applied to data in core memory can be applied equally well to data in peripheral device registers. Therefore peripheral device registers may be manipulated as flexibly as core memory by the central processor.

This is an especially powerful feature, considering the special capability of PDP-11 instructions to process data in any memory location as though it were an accumulator.

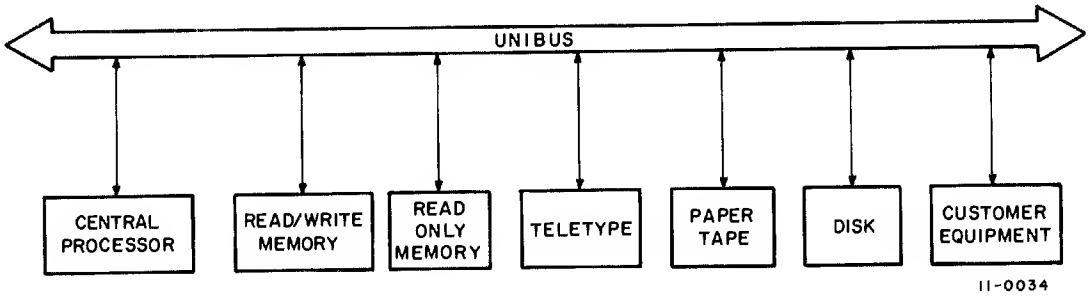


Figure 1-1. PDP-11 System Simplified Block Diagram

1.3.2 Bidirectional Lines

Unibus lines are bidirectional; therefore, the input lines can also be driven as output lines. This means that a peripheral device register can be either read or changed by the central processor or other peripheral devices, and the same data lines can be used for transfer operations. Thus, the same register can be used for both input and output functions.

1.3.3 Master-Slave Relation

Communication between two devices on the bus is in a master-slave relationship. During any bus operation, one device has control of the bus. This device, the bus master, controls the bus when communicating with another device on the bus, called the slave. A typical example of this relationship is the processor, as master, fetching an instruction from memory (which is always a slave). Another example is the disk, as master, transferring data to memory, as slave. Master-slave relationships are dynamic. The processor, for example, passes bus control to a disk. The disk, as master, then communicates with a slave memory bank.

The Unibus is used by the processor and all I/O devices; thus, a priority structure determines which device obtains control of the bus. Consequently, every device on the Unibus capable of becoming bus master has an assigned priority. When two devices, which are capable of becoming a bus master, simultaneously request use of the bus, the device with the higher priority receives control first.

1.3.4 Interlocked Communication

Communication on the Unibus is interlocked between devices. Each control signal issued by the master device must be acknowledged by a response from the slave to complete the transfer. Therefore, communication is independent of the physical bus length and the response time of the master and slave devices. The maximum typical transfer rate on the Unibus is one 16-bit word every 750 ns, or 1.3 million 16-bit words per second.

1.4 PERIPHERAL DEVICE ORGANIZATION AND CONTROL

Registers in peripheral devices are assigned addresses similar to memory; thus, all PDP-11 instructions that address memory locations can become I/O instructions. Data registers in devices can take advantage of all the

arithmetic power of the processor. The PDP-11 controls devices differently than most computer systems. Control functions are assigned a register address, and then the individual bits within that register can cause control operations to occur. For example, the command to make the paper-tape reader read a frame of tape is provided by setting a bit (the reader enable bit) in the control register of the device. Instructions such as MOV and BIS may be used for this purpose. Status conditions are also handled by the assignment of bits within this register, and the status is checked with TST, BIT, and CMP instructions. In addition, there is no limit to the number of registers that a device may have, providing an unlimited flexibility in the design and control of peripheral equipment.

1.5 TRANSFER OF BUS MASTER

A device (other than the central processor) that is capable of becoming bus master generally requests use of the bus for one of two purposes:

a. to make a non-processor transfer of data directly to or from memory, or

b. to interrupt program execution and force the processor to branch to a specific address where an interrupt service routine is located.

Subsequent paragraphs discuss priority structure and the two main uses of the bus: nonprocessor transfers and interrupts.

1.5.1 Priority Structure

When a device capable of becoming bus master requests use of the bus, the handling of that request depends on the location of that device in the priority structure. The following factors must be considered to determine the priority of the request:

a. The processor's priority is set under program control to one of eight levels using bits 7, 6, and 5 in the processor's status register. These three bits set a priority level that inhibits granting of bus requests on the same or lower levels.

b. Bus requests from external devices can be made on any one of five request lines. A nonprocessor request (NPR) has the highest priority, and its request is granted by the processor between bus cycles of an instruction execution. Bus request 7 (BR7) is the next highest priority and bus request 4 (BR4) is the lowest. The four lower level priority requests (BR7 to BR4) are granted by the processor between instructions. When the processor priority is set to a specific level, all bus requests on that level and below are ignored. For example, if the processor priority is 6, requests on BR6 or any other lower level are not granted.

c. When more than one device is connected to the same bus request line, the device physically nearer the processor has a higher priority than the device further away. Any number of devices can be connected to a specific BR or NPR line.

When a device other than the processor gains control of the bus, it uses the bus to perform either a data transfer or an interrupt request as described in the following paragraphs.

1.5.2 Data Transfer

Direct memory or device access data transfers can be accomplished between any two peripherals without processor supervision. These are called NPR level data transfers. Normally, NPR transfers are made between the memory and a mass storage device, such as a disk.

During NPR transfers, it is not necessary for the processor to transfer the information between the memory and the mass storage device. The bus structure enables device-to-device transfers, thereby allowing customer-designed peripheral controllers to directly access other devices (such as disks) on the bus. This direct access capability permits operations such as a disk directly refreshing a CRT display.

An NPR device provides extremely fast access to the bus and can transfer data at high rates once it gains control. The processor state is not affected by this type of transfer; therefore, the processor can relinquish bus control while an instruction is in progress. This release of the bus can normally occur at the end of any bus cycle, however, the bus can never be released between cycles of a read-modify-write sequence. (This is described more fully in Chapter 2.) In PDP–11/10 and 11/20 Systems containing a KA11 Processor, an NPR device gains bus control in 3.5 μs or less. An NPR device in control of the bus transfers 16-bit words to memory at the same speed as the memory cycle time. In the case of the MM11-E Core Memory, transfer occurs every 1.2 μs.

1.5.3 Interrupt Requests

Devices that gain bus control with one of the bus request lines (BR7, BR6, BR5, BR4) can take full advantage of the power and flexibility of the processor by requesting an interrupt. The entire instruction set is then available for manipulating data and status registers. When a device servicing program is to be run, the task being performed by the central processor is interrupted, and the device service routine is initiated. After the device request has been satisfied, the processor returns to its former task. Note that interrupt requests can be made only if bus control has been gained through a BR priority level. An NPR level request can never be used for an interrupt request.

1.5.4 Interrupt Procedure

This paragraph provides an example of an interrupt operation. Assume that a peripheral requires service and requests use of the bus at one of the four BR levels. The operations required to service the device are as follows:

a. Priorities permitting, the processor relinquishes bus control to the device.

b. When the device gains control of the bus, it sends the processor an interrupt command and a unique address of a memory location which contains the starting address of the device service routine. (This is called the interrupt vector address.) Immediately following this pointer address is a word (located at vector address +2) to be used as the new processor status (PS) word.

c. The processor pushes the current central processor status word and then the current program counter (PC) value on the processor stack. The stack is pointed to by register R6.

d. The new PC and PS (the interrupt vector) are taken from the address specified by the device, and the device service routine is initiated.

NOTE

These operations are performed automatically and no device polling is required to determine which service routine to execute.

e. These operations are performed in 7.2 μs from the time the central processor receives the interrupt command until it fetches the first instruction of the service routine. Note that this time interval assumes that no NPR transfers occurred during this time.

f. The device service routine can cause the processor to resume the interrupted process by executing the return from interrupt (RTI) instruction which pops the two top words from the processor stack and transfers them back to the PC and PS registers. This instruction requires 4.5 μs, provided there are no intervening NPR requests.

g. A device service routine can, in turn, be interrupted by a higher priority bus request any time after the first instruction of the routine has been executed.

h. If such an interrupt occurs, the PC and PS of the current device service routine are automatically pushed onto the stack, and the new device routine is initiated as before. This nesting of priority interrupts can continue to any level; the only limitation is the amount of core memory available for the processor stack.

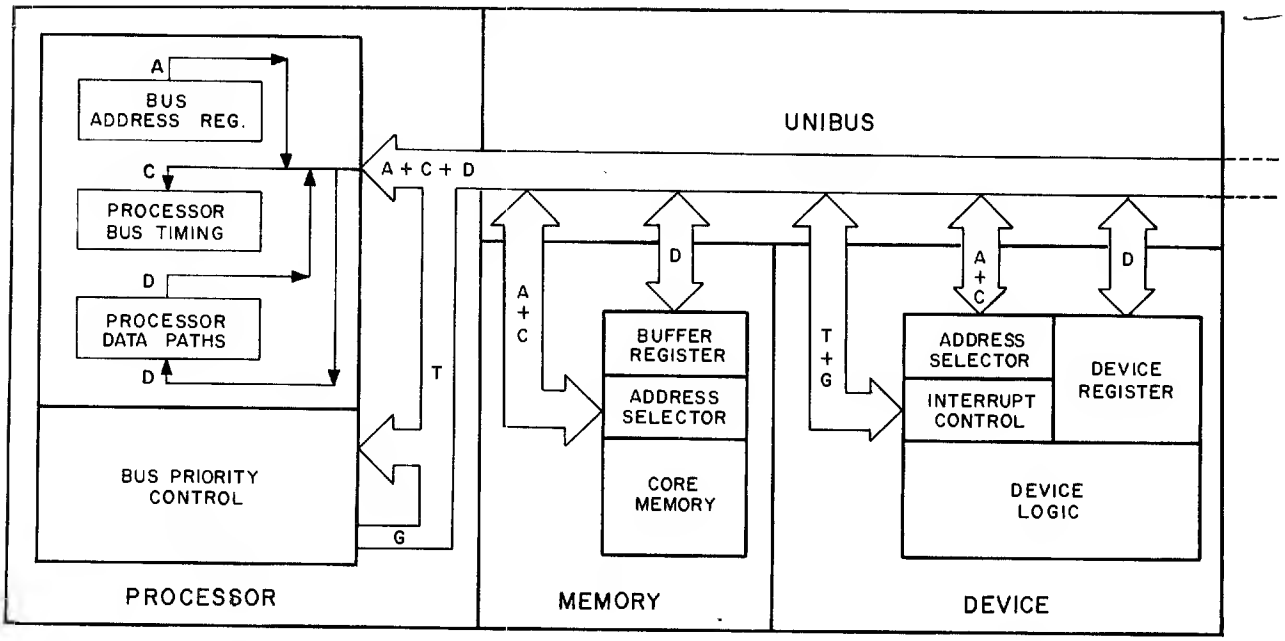
Chapter 2
Unibus Theory and Operation

2.1 INTRODUCTION

This chapter provides detailed theory of Unibus operation and includes descriptions of bus signal lines, bus transactions, bus timing, and bus interfacing characteristics. In addition, this chapter contains discussions of address allocations and device registers.

2.2 UNIBUS SIGNAL LINES

The PDP-11 Unibus consists of 56 signal lines. All devices, including the processor, are connected to these lines in parallel (see Figure 2-1). The bidirectional nature of 51 signal lines permits signals to flow in either direction. The remaining five unidirectional lines are used for priority bus control. All 56 signals and their functions are listed in Table 2-1 and described in subsequent paragraphs. In addition, Unibus pin assignments are listed in Appendix C.



KEY A-ADDRESS INFORMATION
C-CONTROL + TIMING SIGNALS
D-DATA INFORMATION
T-CONTROL TRANSFER SIGNALS
G-BUS GRANT SIGNALS

Figure 2-1 PDP-11 Interface Block Diagram

Table 2-1
Unibus Signals

Name	Mnemonic	Source	Destination	Timing	Function
Data Transfer Signals (For transfer of data to or from master)					
Address	A(17:00)	Master	All	MSYN	Selects slave device
Data	D(15:00)	Master	Slave	MSYN (DATO, DATOB)	
		Slave	Master	SSYN (DATI, DATIP)	
Control	C(1:0)	Master	Slave	MSYN	Selects transfer operation
Master Sync	MSYN	Master	Slave	Beginning of transfer	Initiates operation and gates A,C and D signals
Slave Sync	SSYN	Slave	Master	Data accepted (DATO, DATOB) Data Available (DATI, DATIP)	Response to MSYN
Parity Available	PA	Master	Slave	Same as Data	Indicates paritied data
Parity Bit	PB	Master	Slave	Same as Data	Transmits parity bit
Priority Transfer Signals (For transfer of bus control to a priority selected master)					
Non-processor Request	NPR	Any	Processor	Asynch-ronous	Highest priority bus request
Bus Request	BR(7:4)	Any	Processor	Asynch-ronous	Requests bus mastership
Non-Processor Grant	NPG	Processor	Next master	In parallel with data transfer	Transfers bus control
Bus Grant	BG(7:4)	Processor	Next Master	After in-struction	Transfers bus control
Selection Acknowledge	SACK	Next Master	Processor	Response to NPG or BG	Acknowledges grant & inhibits further grants
Bus busy	BBSY	Master	All	Except during transfer of control	Asserts bus mastership
Interrupt	INTR	Master	Processor	After asserting BBSY (not after NPR) device may perform several transfers before asserting INTR.	Transfers bus control to handling routine in processor

Table 2-1.
Unibus Signals (cont)

Name	Mnemonic	Source	Destination	Timing	Function
Miscellaneous Signals					
Initialize	INIT	Processor	All	Asynch-ronous	Clear and reset signal
AC line low	ACLO	Power	All	Asynch-ronous	Indicates impending power failure
Spare	SP				Spare, Unused in 11/10, 11/20

2.2.1 Data Transfer Lines

Forty bidirectional bus lines are used for data transfer. In a data transfer, one device is bus master and controls the transfer of data to or from a slave device. The processor is always bus master when no other device is using the bus, and it is master for all data transfers involved in normal instruction processing.

Data transfer signals are listed in Table 2-2.

Table 2-2
Data Transfer Signals

Name	Mnemonic	No. of Lines
DATA	D⟨15:00⟩*	16
ADDRESS	A⟨17:00⟩	18
CONTROL	C⟨1:0⟩	2
MASTER SYNC	MSYN	1
SLAVE SYNC	SSYN	1
PARITY AVAIL.	PA	1
PARITY BIT	PB	1
		TOTAL: 40

* Throughout this manual, the notation A⟨a:b⟩ specifies a-b+1 signal lines, which are named Aa through Ab.

Simplified and standardized control logic is made possible by using separate dedicated lines for all signals. In any data transfer, data is transmitted and received; the master device provides the address of the slave device; and control and timing signals are provided. Each of these three functions occurs on a distinct set of bus lines, eliminating the use of additional hardware and extra timing states to distinguish between address, control information, and data.

2.2.1.1 **Data Lines (D⟨15:00⟩)** – The 16 data lines are used to transfer information between master and slave. The bit format is shown in Figure 2-2.

2.2.1.2 **Address Lines (A⟨17:00⟩)** – The 18 address lines are used by the master device to select the slave (a unique memory or device register address) with which it will communicate. The reason for 18 address lines is to

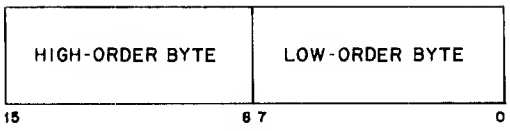


Figure 2-2. Data Line Bit Format

extend the total memory capability of future members of the PDP-11 family to 262,144 bytes. The bit format of the 18 signals is shown in Figure 2-3.

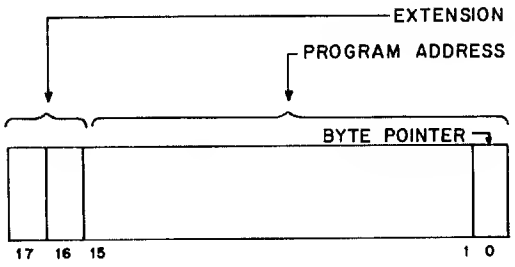


Figure 2-3. Address Line Bit Format

Lines A(17:01) specify a unique 16-bit word. In byte operations, A00 specifies the byte being referenced. If a word is referenced at X (X must be even, since words can be addressed on even boundaries only), the low-order byte can be referenced at X and the high-order byte at X+1.

Only 16 bits are normally supplied by programs as memory reference addresses. In the KA11 Processor used with PDP-11/10 and PDP-11/20 Systems, lines A17 and A16 are asserted (forced to 1) whenever the processor attempts to reference an address between 160000 and 177777; i.e., where A15=A14=A13=1. Thus, the processor converts this 16-bit address to a full 18-bit bus address.

Peripheral devices are normally assigned an address from within the bus address allocations from 760000-777777 (program addresses, 160000-177777).

2.2.1.3 Control Signals – The control signals are divided into three groups: signals that select data transfer operations, signals that allow the master and slave device to communicate, and signals used for parity checking.

a. Control Lines (C(1:00)) – These two bus signals are coded by the master device to control the slave in one of four possible data transfer operations shown in Table 2-3.

b. Master and Slave Synchronization – Master synchronization (MSYN) is a control signal used by the master to indicate to the slave that address and control information is present. Slave synchronization (SSYN) is the slave's response to MSYN.

c. Parity Available (PA) and Parity Bit (PB) – These signals are for devices on the Unibus that use parity checks. PA indicates that the data being transferred is to use parity, and PB transmits the parity bit. Neither line is used by the KA11 Processor.

Table 2-3
Data Transfer Operations

C1	C0	Operation
0	0	DATI-Data In
0	1	DATIP-Data In, Pause
1	0	DATO-Data Out
1	1	DATOB-Data Out, Byte

2.2.2 Priority Transfer Lines

The Unibus contains 13 lines classified as priority transfer lines. Five of these are priority bus request lines (BR⟨7:4⟩, NPR) and five are the corresponding grant lines (BG⟨7:4⟩, NPG) which the processor uses to respond to a specific bus request. Each device of the same priority level passes a grant signal to the next device on the line, unless it has requested bus control; in this case, the requesting device blocks the signal from the following devices and assumes bus control. A discussion of physical chaining of devices to create priority levels is presented in Paragraph 2.4.3. In addition, there are three other control lines: SACK, BBSY, and INTR. All 13 lines are described below.

a. Bus Request Lines (BR⟨7:4⟩) – These four bus signals are used by peripheral devices to request control of the bus.

b. Bus Grant Lines (BG⟨7:4⟩) – These signals are the processor's response to a bus request. They are asserted only at the end of instruction execution, and in accordance with the priority determination.

c. Non-processor Request (NPR) – This signal is a bus request from a peripheral device to the processor.

d. Non-Processor Grant (NPG) – This signal is the processor's response to an NPR. It occurs at the end of a bus cycle within the instruction execution.

e. Selection Acknowledge (SACK) – SACK is asserted by a bus-requesting device that has received a bus grant. Bus control passes to this device when the current bus master completes its operation.

f. Interrupt (INTR) – This signal is asserted by the bus master to start a program interrupt in the processor.

g. Bus Busy (BBSY) – This signal is asserted by the master devices to indicate bus use.

2.2.3 Miscellaneous Control Lines

There are three additional lines on the Unibus which may be used by all devices. These lines are: initialize, ac line low, and a spare line. In addition, one signal is available for devices built on system units that are mounted within a BA11 Mounting Box and powered by an H-720 Power Supply. This signal is dc line low (DCLO) which is not a Unibus signal but may be useful in certain devices.

a. Initialization (INIT) – This signal is asserted by the processor when the START key on the console is depressed, when a RESET instruction is executed, or when the power fail sequence occurs. In the latter case, INIT is asserted following the power fail service routine while power is going down, and again when power comes up. INIT may also be used to clear and initialize peripheral devices.

b. AC Line Low (ACLO) – This is an anticipatory signal which starts the power fail trap sequence, and may also be used in peripheral devices to terminate operations in preparation for power loss. When ACLO is cleared, the power up instruction sequence in the processor begins.

c. Spare (SP) – This line is not currently used in the KA11 Processor contained in PDP-11/10 and PDP-11/20 Systems.

d. DC Line Low (DCLO) – This is not a Unibus signal but is available from the H720 System Power Supply. This signal is wired to the power connector card, which is plugged in system unit power slot A3 (refer to Table C-3). This signal remains cleared as long as all dc voltages are within specified limits. If an out-of-voltage condition occurs, DCLO is asserted by the power supply. Devices such as memories use the DCLO signal to

inhibit further operations. The DCLO signal is always cleared before ACLO when power is coming up and is asserted after ACLO when power is going down.

2.3 DATA TRANSFER BUS TRANSACTIONS

All bus activity is asynchronous and depends on interlocking of control signals. In every case, a signal from a slave device is generated in response to a signal from a master device, and the master signal is dropped in response to the slave signal. The complete elimination of critical self-timing enables the bus to operate with various devices of a wide range of speeds.

NOTE

Signals on the Unibus are asserted when *low* (except for the unidirectional bus grant lines). All timing diagrams in this manual reflect the asserted and cleared levels of Unibus lines as described in Paragraph 3.2.2.

The four bus-data transfers are described in Table 2-4. The bus master determines one of the four data transfers by asserting the proper code on the C⟨1:0⟩ lines.

NOTE

All data transfers are with reference to the master device; data-in is always from slave to master, and data-out is from master to slave. For example, when the processor (master) loads data into the core memory (slave), a data-out bus operation is performed.

Table 2-4
Bus-Data Transfer Transactions

Name	Mnemonic	C Lines C1	C Lines C0	Function	Octal Code
Data in	DATI	0	0	Data from slave to master	0
Data in, pause	DATIP	0	1	Inhibits restore cycle in destructive read-out devices; Pause flip-flop is set which inhibits clear cycle on following DATO (B). Must be followed by DATO or DATOB.	1
Data out	DATO	1	0	Data from master to slave.	2
Data out, Byte	DATOB	1	1	Transfers data from master to a single byte in slave. Data transmitted on D⟨15:08⟩ for A00=1 D⟨07:00⟩ for A00=0.	3

The DATI and DATIP transactions request transfer of data from a slave, the address of which is specified by A⟨17:00⟩, to the master. Both transfers use the data lines to carry the data. There is no distinction made by the slave as to whether the transfer is used for byte or word data. The slave places the data on D⟨15:00⟩. It is the function of the master device to retrieve the data from the proper lines: low-order byte register with A00=0 from D⟨07:00⟩; high-order byte register with A00=1 from D⟨15:08⟩; or word register from D⟨15:00⟩. The DATIP operation is identical to the DATI, except DATIP is used to inform the slave device that this is the first part of

an in-modify-out cycle. A DATIP normally sets a pause flag in the destructive read-out device (i.e., core memory) which inhibits the restore cycle. The DATIP must be followed by a data-out cycle (DATO or DATOB), and the master must retain bus control until it is completed. In nondestructive readout devices (i.e., flip-flops), the DATI and DATIP are treated identically.

The DATO and DATOB operations transfer data from the master to the slave. A DATO is used to transfer a word to the address specified by A⟨17:01⟩. The slave ignores A00 and the master places data on D⟨15:00⟩. A DATOB is used to transfer a byte of data to the address specified by A⟨17:00⟩. Line A00=0 indicates the low-order byte, and the master places the data on lines D⟨07:00⟩.

Appendix B lists all the bus cycles used by the KA11 Central Processor during the execution of instructions.

2.3.1 Data Transfer Timing

The design of the Unibus imposes certain timing restrictions although transfers are interlocked. Responsibility for these timing restrictions has been assigned to the master to simplify the slave design.

In all transfers, it is assumed that there can be a maximum 75-ns skew due to driver, receiver, and transmission line tolerances. In other words, the coincident assertion of two lines at the transmitter inputs of one device could result in a maximum difference of 75-ns in the occurrence of those signals at the receiver outputs in another device.

Because of this possible skew, the master always delays its MSYN signal to ensure that MSYN does not reach the slave device prior to valid data or addresses. In addition, the MSYN signal is further delayed to allow 75 ns for decoding by the slave device. The master also must not drop the A (address) or C (control) lines until 75 ns after MSYN has been dropped to guarantee that there are no spurious selections. Note, however, that when a slave transmits data to a master (DATI or DATIP), the deskew and decode time delay must be made by the master (refer to Paragraph 2.3.2 f). Additional timing information can be found in Paragraph 2.5.

2.3.2 DATI and DATIP Bus Transactions (See Figure 2-4)

All data transfer functions are with reference to the master device; therefore, data-in (DATI) indicates data transfer from the slave to the master.

a. The master device places the address (or location) of the slave device on the A lines, and places 0 on the C lines for a DATI, or 1 for a DATIP. The master device then waits 150 ns (minimum): 75 ns to allow for worst-case signal skew, and 75 ns to allow internal logic in the slave devices to decode the address.

b. The slave device decodes the address and control bits to determine if the slave is to participate in a data transfer.

c. If the slave device from a previous bus transaction still has SSYN asserted, the master device waits until SSYN is clear. MSYN is asserted when SSYN is clear and the delay from step *a* is complete.

d. When the MSYN signal is received by the slave device, the device prepares the data for transmission to the master. For devices such as core memory, this means performing a read cycle. For flip-flop registers, the data is available immediately.

e. When data is available, the slave places data on the D lines and asserts SSYN. If the slave is a destructive read-out device, it enters a restore cycle, if the command was a DATI; for a DATIP, the slave sets a pause flag and waits for modified data before performing a write cycle.

f. The master device receives SSYN and the data. After a 75-ns minimum delay to allow for skew and time for internal gating, the master device strobes the data and clears MSYN.

g. After a 75-ns minimum wait, to ensure that no spurious device selections occur as various bits of the address change while MSYN is still asserted, the A and C lines are cleared.

h. When MSYN is cleared, the slave clears the D lines and SSYN; the bus is now free for other use.

i. The master receives the cleared SSYN, which signals the end of the current bus transaction. If an

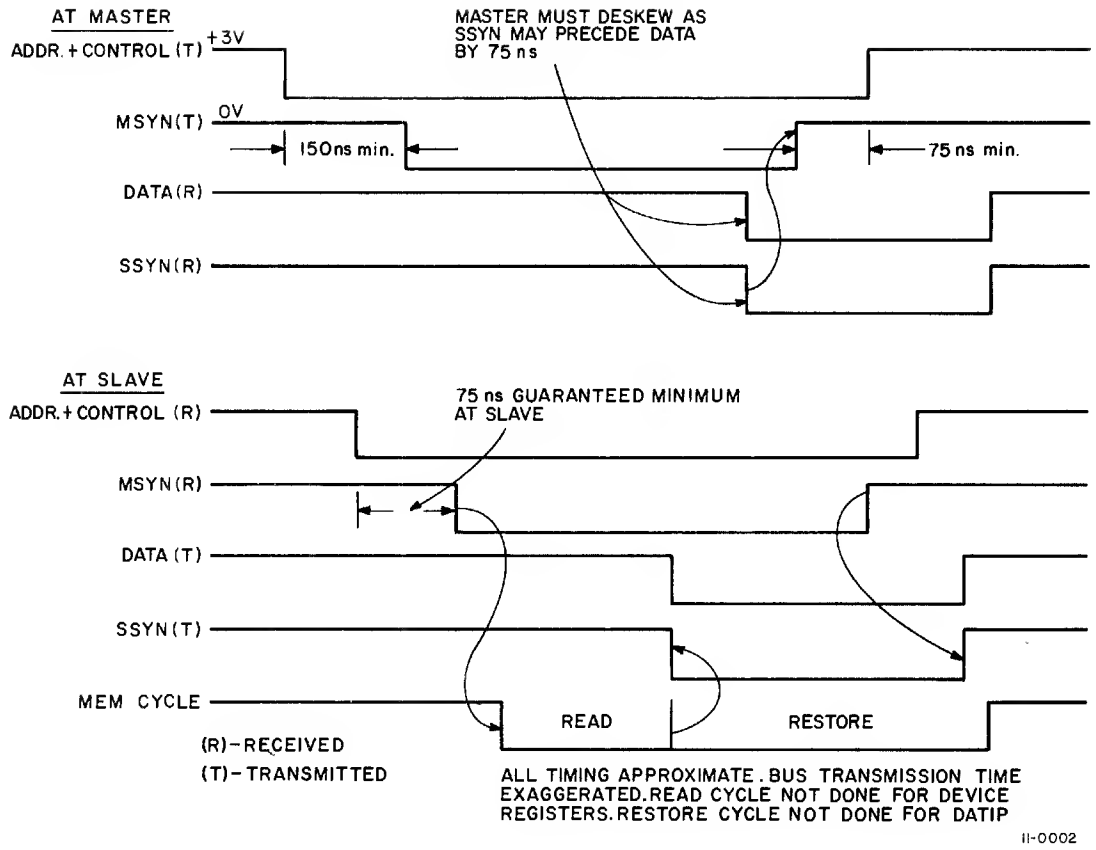


Figure 2-4. DATI and DATIP Timing Diagram

output transfer follows (a DATO or a DATOB must follow a DATIP), step *a* of the DATO(B) can start now. If another input transfer follows, it can start after step *g* is complete; however, MSYN cannot be asserted until the conditions of step *c* are met.

2.3.3 DATO and DATOB Bus Transactions (See Figure 2-5)

Because all data transfers are with reference to the master device, a data out (DATO) indicates data transfer from the master to the slave.

a. The master device places the address (or location) of the slave device on the A lines, the data on the D lines, and asserts 2 on the C lines for a DATO or asserts 3 on the C lines for a DATOB.

b. After a 150-ns minimum wait (75 ns to allow for worst case signal skew, and 75 ns to allow internal logic in the skew device to decode the address) the master device asserts MSYN if the bus is inactive (SSYN is clear).

c. The slave decodes A(17:00) and, if this is the assigned address of the slave, it responds to MSYN by taking in the data and then asserting SSYN. For a DATO, the slave accepts a full word; for a DATOB, the slave accepts one byte as determined by A00. For consecutive operations with the same slave device, the slave may have to complete an internal action (such as a restore or write cycle) before responding. If the slave is a destructive readout device with the pause flag set (indicating the previous bus transaction was a DATIP), the slave immediately begins a write cycle.

d. The master device receives SSYN and clears MSYN.

e. After a 75-ns minimum wait, the master clears the A, D, and C lines.

f. The slave device receives cleared MSYN and clears SSYN.

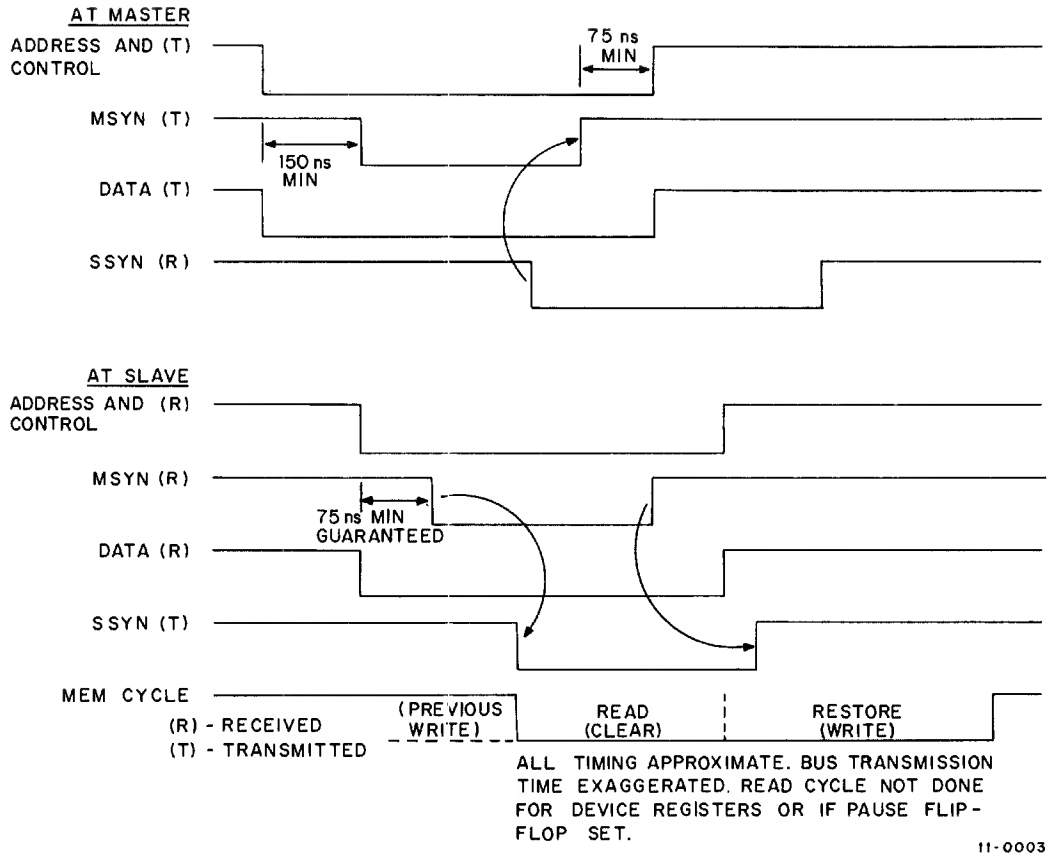


Figure 2-5. DATO or DATOB Timing Diagram

g. The master receives cleared SSYN which signifies the end of the bus transaction. A new bus transaction can begin at step e of a DATO or DATOB, and the master can assert MSYN at step g, when SSYN is cleared.

2.4 PRIORITY TRANSFER TRANSACTIONS

Transfer of bus control from one device to another is determined by priority arbitration logic, which is part of the KA11 Central Proeessor. Requests for the bus can be made at any time (asynchronously) on the bus request (BR) and non-processor request (NPR) lines. During each bus cycle, the arbitration logic first checks for an NPR request (since these requests always take precedence over processor use of the bus). If an NPR is present, the logic issues an NPG signal and receives a selection acknowledge (SACK) signal in return. This procedure occurs simultaneously with the current data transfer. When the device scheduled to become the new bus master is selected, it waits for the present master to clear bus busy (BBSY); then, the newly selected device becomes bus master and asserts BBSY.

A similar procedure occurs at the end of each instruction when the priority arbitration logic checks the bus request lines against the processor priority (as determined by bits ⟨07:05⟩ of the processor status register) and the priority logic issues a grant on the corresponding line. Thus, one of the four levels of BR requests is granted by the processor between instructions unless the instruction currently being executed causes an internal trap (either

an error or trap instruction). In this case, BR requests are not granted until completion of the first instruction following the trap sequence. The highest request is always granted first (if the processor priority level is lower than the request level). The grant signals always pass serially through each device connected to the corresponding level in the system. If a device makes a request, it blocks the signal transmission to the next device on the line; otherwise, it passes the signal on.

This causes the device closest to the processor to be the highest subpriority on each request level.

2.4.1 **Priority Transfer (PTR) Transaction** (See Figure 2-6)

The signal sequence by which a device becomes selected as next bus master is the priority transfer (PTR) bus operation. This operation does not actually transfer bus control; it only selects a device as next bus master. The sequence of events is as follows:

a. The device which needs a data transfer asserts the BR (or NPR) line assigned to it.

b. The processor receives one or more BR signals. These signals enter a priority arbitration system, which compares the NPR and BR levels with the processor priority levels and with each other. If a request has the highest priority entering the arbitration system, and the SACK line is clear, the processor asserts the corresponding BG (or NPG) line. NPG is asserted during the current bus transaction, while BG is asserted only at the end of the current instruction.

c. Each device on the asserted BG line passes the BG signal, unless it is requesting bus control.

d. The first device on the line which has BR asserted responds to the BG by asserting SACK, blocking the BG signal from following devices, and clearing BR.

e. The processor receives the SACK signal and clears BG.

f. The current bus master completes a data transfer and clears BBSY at the same time it clears MSYN.

g. The selected device, which is the new bus master, asserts BBSY when BBSY, BG, and SSYN are clear at the end of the previous data transfer. INTR may be asserted at this time, if the new bus master is interrupting. (Refer to Paragraph 2.4.2.)

h. SACK is dropped at the same time INTR is asserted if the device is interrupting. If the device is transferring data, the SACK signal is dropped just prior to the start of the last bus cycle that the device uses.

i. When the new bus master has completed its last data transfer, it clears BBSY. A new bus master then takes control of the bus. If no device is selected (SACK is clear), the processor asserts BBSY and continues processing. If, instead of clearing BBSY in a passive release, the device asserts INTR, the processor conducts an INTR bus transaction. This is called an active release of the bus.

NOTE

Since an NPR is granted within an instruction, and the interrupt and following processor response would destroy information held in the processor, devices granted bus control through an NPR must not cause a processor interrupt (refer to Paragraph 2.4.2).

2.4.2 **Interrupt Transaction** (See Figure 2-7)

A device may cause the interrupt operation to occur any time it gains bus control with one of the BR levels. It is usually accomplished immediately on becoming bus master; however, it may follow one or more data transactions on the bus.

a. A device which has been selected as bus master asserts INTR and a vector address on the D lines, at the same time that it clears SACK and asserts BBSY. SACK must remain asserted until INTR is asserted. If the device has been making data transfers prior to the interrupt, it should assert SACK through the last cycle.

b The processor receives the INTR signal, waits 75 ns for deskew to ensure that all bits of the interrupt vector address are available, and asserts SSYN when the data is read in.

c. The bus master (interrupting device) receives SSYN and clears INTR, the D lines, and BBSY. This constitutes active release of the bus to the processor.

d. The processor clears SSYN when INTR is cleared, and enters the interrupt sequence to store the

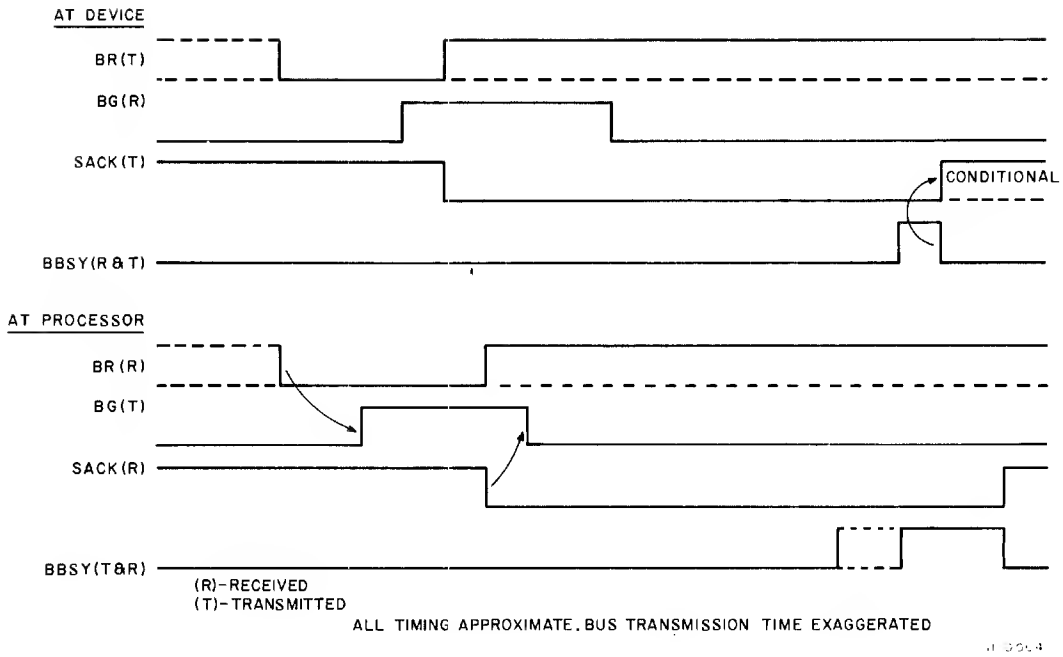


Figure 2-6. PTR Timing Diagram (nominally, for processor master)

contents of the current PC and PS registers and replace them with the contents of the location specified by the vector address.

2.4.3 Priority Chaining

The PDP-11 uses physical chaining of devices to assign minor priority levels. These levels separate devices of the same major priority level to provide a full array of priority servicing. Figure 2-8 illustrates the mode of operation and advantages of this system. Six devices are shown in order of their physical (or electrical) distance from the processor. Three devices are at major priority level 4: device A, device C, and device D. The remaining three are at major priority level 5.

If the processor is at priority level 5 or above, no bus requests are granted from any of these devices. At a processor priority of 4, only requests from devices B, E, or F are granted. Assume that the processor priority is 2 and also that during one instruction cycle, devices C, E, and F assert bus requests. At the end of the instruction, the processor conducts a PTR operation. Since BR 5 is asserted, the processor does not respond to BR 4 (device C). When BG 5 is asserted, the signal first goes to device B. After a delay to clock the interrupt control shift register (discussed in Paragraph 3.3.5), the signal is passed on, since device B was not asserting BR and does not block the pulse. Next, the signal goes to device E, which blocks the pulse, drops BR 5, and takes control of the bus. Device F still has BR 5 asserted, however, and device C has BR 4 asserted. These requests remain on the bus until granted or actively cleared by the processor. If device E does an INTR operation, device F gains control of the bus after the first instruction of the handling routine has been executed, unless the INTR operation raises the processor priority.

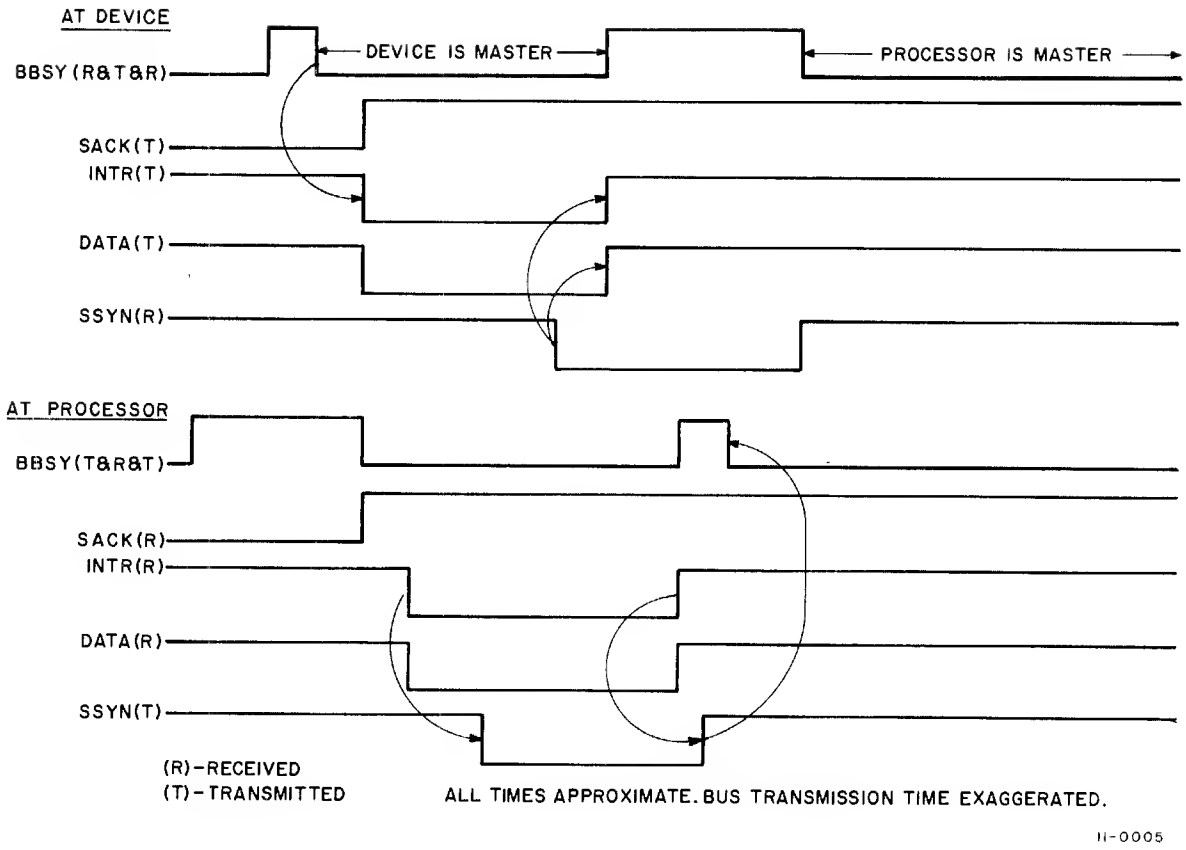


Figure 2-7. INTR Timing Diagram

Changing the processor priority is accomplished easily since the INTR operation provides a new PS word, which includes a new processor priority. If the priority is set to 5, the processor ignores the current bus requests but grants requests from other devices with higher major priority levels (if there are any).

At the conclusion of the interrupt handling routine, the original processor priority is restored and normal processing is resumed. After one instruction, device F gains control of the bus. When normal processing resumes again, device C, which is still waiting for bus service, gains control in a similar manner.

Higher priorities are assigned to devices that require faster service to avoid destruction or loss of data. Slower devices, which can afford to wait, operate with low priorities. Therefore, service can be provided to all devices in an equitable manner, with no lost data and maximum speed and bus efficiency.

2.5 UNIBUS TIMING

Although all bits of an information signal are transmitted simultaneously, differences in bus path lengths and speeds of individual gate responses may cause variations in transmission time and in the elapsed time before reception. To allow for slow signals to arrive, and to permit settling of levels which have encountered transmission noise, the strobing or gating of this data is delayed a nominal 75 ns. This delay is greater than the worst case signal skew encountered in practice.

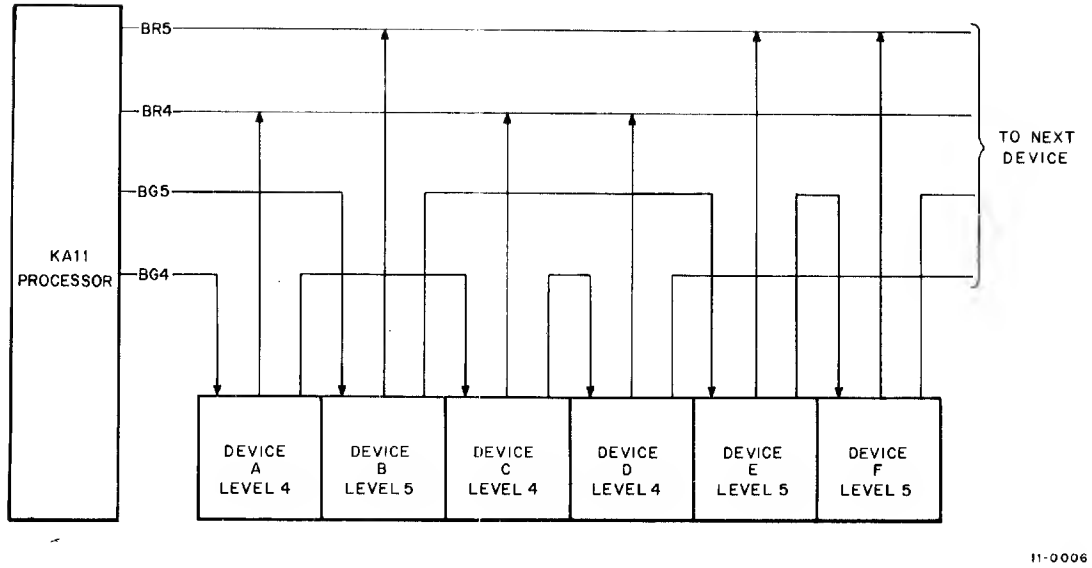


Figure 2-8. Priority Chaining Example

A further delay may be necessary to allow an information signal within a device to qualify gates that accept a strobing signal. A 75-ns delay allows for this gating and must be provided by any device which acts as bus master for a data transfer. Thus, a slave is always guaranteed that address and data are valid at its interface (the device side of the receivers) 75 ns in advance of the MSYN signal at the output of the MSYN receiver. If a slave requires more decoding time, it must provide its own delay for the MSYN signal, or trigger a delayed strobe from the MSYN signal.

To simplify slave device design in a DATI or DATIP sequence, the slave may place the data on the D lines coincident with the assertion of SSYN. The deskewing (75 ns) and decoding delay is the master's responsibility. In the INTR sequence, the interrupting device may place the vector address on the D lines coincident with the INTR signal. The processor allows for the 75-ns skew.

2.5.1 Timing Example

To illustrate the operating speed of a data transfer, assume that a DATI is to be performed for a slave device that has data stored in a flip-flop register. A typical transmitter-bus-receiver delay is 75 ns. The transfer procedure is shown in Figure 2-9 and described below.

The bus master places address and control information on the bus, waits 150 ns, and then asserts MSYN. After 75 ns, the slave device recognizes MSYN and takes an internal delay of 70 ns before gating data into the bus transmitters. The master waits for the transmitter-receiver propagation delay (75 ns) and a 75-ns deskew time before it strobes the data. An additional 30 ns is required to clear MSYN. After another 75-ns propagation delay, a 30-ns delay to clear SSYN, and a final propagation delay, the cycle is complete. The total cycle time is 655 ns. Note that a second DATI cycle can start when the address and control lines are cleared by the master. The only

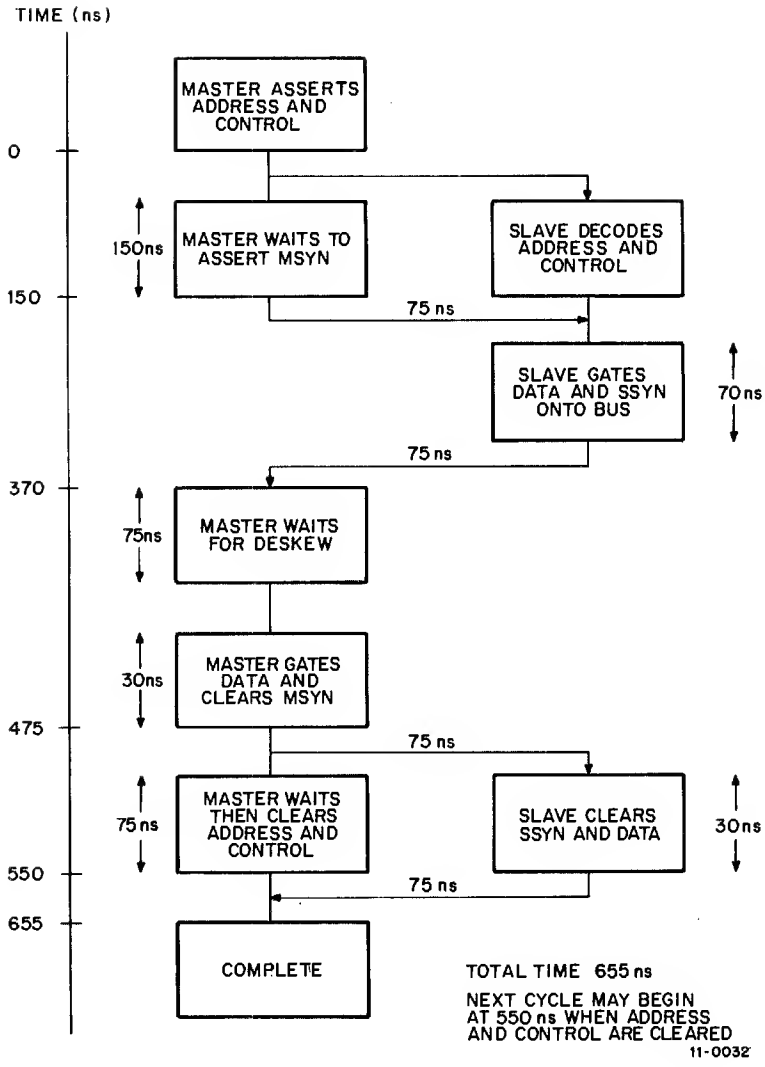


Figure 2-9. Typical DATI Timing Flow

restriction is that SSYN must be clear before MSYN is asserted; with the times indicated, this would occur. Because of overlap, repetitive cycles take only 550 ns.

If the next cycle is a DATO, however, the master waits 150 ns from the time the data lines are loaded with the proper data required for the DATO, until MSYN is asserted. If the master asserts the new data after 550 ns, it may overlap the slave data from the previous cycle. This causes no problem, but the data lines do not reflect the correct data until the master sees SSYN cleared. At this time, it can start the 150-ns delay needed prior to asserting MSYN.

2.5.2 Time-Out Protection

A precaution must be taken when designing peripheral devices that gain control of the bus for the purpose of transferring data to another element on the Unibus. Normally, such a device contains a bus address register,

which is loaded by the program as one of the initialization steps. This address must then be incremented by the device upon completion of each data transfer. If the program loads an erroneous address or if the register increments beyond the available core memory in the existing system, no SSYN response is generated for the data transfer. To prevent this problem from hanging up the system, it is recommended that a 5- to 10-μs integrating one-shot be triggered each time the master device asserts MSYN. If this one-shot times out before SSYN is received, the master should stop the transfer by clearing MSYN, BBSY, and any other signals it has asserted. The master should then set an error flag in its status register.

2.6 ADDRESS MAPPING

A PDP-11 Address Map is shown in Figure 2-10. Observe that, in the following discussion, all addresses are numbered in octal. The letter K, which is normally used to denote 1000_{10}, is used in this discussion to denote 1024_{10} (2^{10}).

The Unibus addresses 2^{18} locations ($262{,}144_{10}$ or 256K), and each location contains eight bits. On the basic PDP-11/10 and 11/20 Systems only 16 bits of address information are under program control. This limits the processor to an address map of 64K locations. Since the word length and bus width in the PDP-11/10 and 11/20 are two bytes, most bus operations access two locations at once; the address supplied on the bus is that of the even-numbered location, and the next higher odd location is selected as well. Byte operations can explicitly address any byte. For example, a DATI to location 400 transfers the information in locations 400 and 401, while a DATOB to location 400 loads only location 400. In all cases, a full-word operation cannot address an odd-numbered location.

The address map (Figure 2-10) contains full, 18-bit wide bus addresses. Hardware in the processor forces A(17:16) to ones if A(15:13) are all ones when the processor is master; thus, the last 8K byte locations are relocated to be the highest locations accessible by the bus. All device addresses and internal processor locations are assigned in these 8K locations.

2.6.1 Interrupt and Trap Vector Locations

The first 400 locations in the address map are reserved for trap and interrupt vectors. The stack pointer overflow feature of the processor protects the data in these locations from destruction if the system stack expands downward into this area. Locations 0 through 37 are used for trap vectors for internal processor use, locations 40 through 57 are reserved for use as system software communications words, and the remaining locations (up to location 400) are used for device interrupt vectors.

To prevent customer-designed interfaces from interfering with standard DEC products, the vector addresses (170, 174, 270, and 274) are reserved for customer interfaces.

Each vector requires four locations, and the vector addresses are constrained to even-word boundaries; that is, each vector must end in 4 or 0. (This is implemented by providing vector addresses which do not specify bits 0 or 1. Since the low bits are always zero, address bit 2 specifies either 0 or 4.)

2.6.2 Memory Locations

Memory locations, either core or ROM, begin at 0 and proceed to 157777. The highest numbered 8K-block in the map is used by device registers and by internal processor register addresses.

2.6.3 Device Register Locations

Each device has one or more device registers. Device register addresses are always even (A00 is 0), although byte operations may address either half of a register.

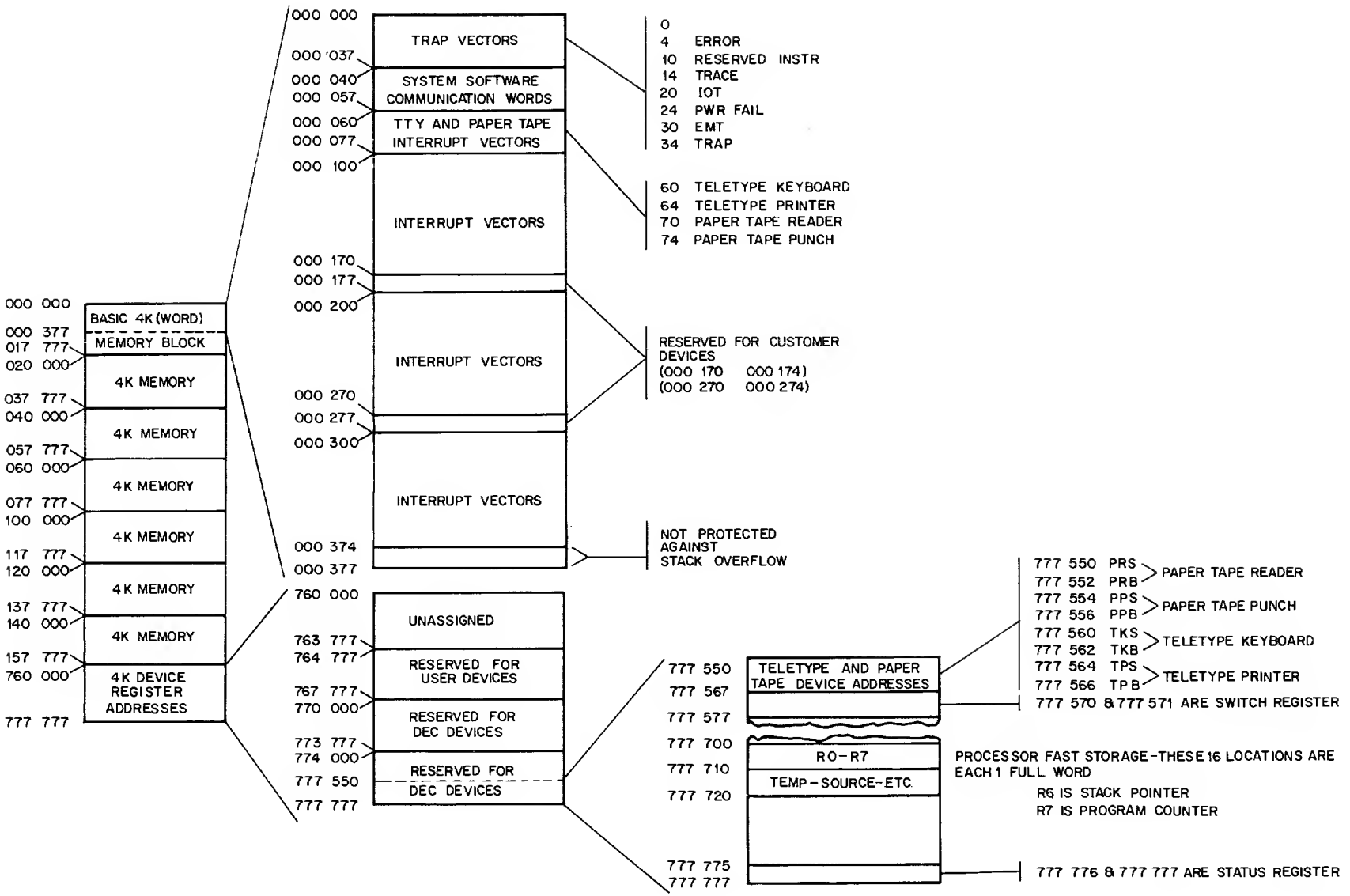


Figure 2-10. PDP-11 Address Map

The top 8K byte locations are allocated for device register assignment. The top 4K byte (770000-777777) is reserved by DEC for processor addresses and standard peripheral devices. The 2K byte addresses (764000-767777) are reserved for customer allocation and are never assigned by DEC. It is recommended that customer-built interfaces be given addresses in this area.

Starting at location 777550, the first eight locations are reserved for use with the first Teletype and first high-speed paper-tape device. Normally, the PDP-11 System is supplied with the Model 33 ASR Teletype or equivalent device. If a high-speed paper-tape device is used, it is a PC-11 Reader Punch. Specific addresses for each register are shown on the address map (see Figure 2-10).

2.6.4 Processor Locations

Only two processor data locations are explicitly addressable. The console switch register, at locations 777570 and 777571, is a read-only switch register on the PDP-11/20 Console and may be used for program controlled data transfers. The processor status register (PS), at locations 777776 and 777777 contains the processor priority level and the condition codes. The 16 processor storage locations (described in Appendix A) are located at the 16 addresses from 777700 to 777717. Each address accesses a full word of data. These registers are not addressable from the bus, and the addresses are used only for deposit and examine functions from the console.

2.7 DEVICE REGISTERS

The actual transfer of data between a device and the Unibus takes place through one or more registers in the device. These registers may be either flip-flop storage registers or dynamic signals which are simply gated to the bus during a transfer. In addition, it is not necessary for the exact nature of the register bits to be the same. Some bits may be used for read/write (transferred on both DATI and DATO transactions); some may be write only (participate only in DATO transactions, and appear as zeros for DATI's); and some may be read only (participate only in DATI's, unaffected by DATO's). Examples of all three types are usually found in control and status registers. A typical example of a read/write bit is an interrupt enable bit; an example of the write bit is a go command bit; and an example of a read bit is an indicator of an error condition requiring operator intervention.

To standardize register format types, DEC has adopted some preferred bit assignments which are shown in Figure 2-11. The preferred order of register address assignments is given in Table 2-5. These preferences are included for reference only and should not be construed as mandatory requirements for interfacing to the Unibus. The exact nature of register assignments varies with each device. The general philosophy of this is illustrated and discussed in Chapter 4.

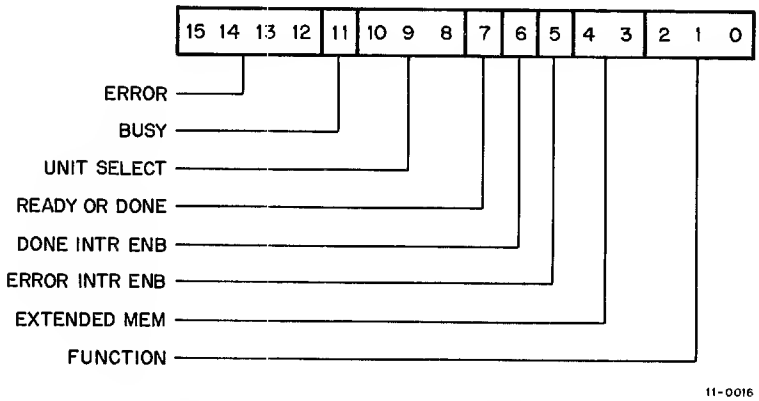


Figure 2-11. Preferred CSR Bit Assignments

FUNCTION: Device Function (read, write, punch, search, etc.) Single function devices should use bit 0 because INC CSR (an operate instruction) effects the command faster than a conventional MOV.

EXTENDED MEM: Used to specify A(17:16) when doing device controlled data transfers to locations not in the first 64K block of addresses.

DONE INTR ENB: Done Interrupt Enable. Inhibits Interrupt on done if not set.

ERROR INTR ENB: Error Interrupt Enable. Inhibits Interrupt on error if not set.

READY or DONE: Bit set by device when internal processing is completed and the device is ready to participate in a transfer. Can be checked by the instruction sequence LOOP: TSTB CRS, BPL LOOP.

UNIT SELECT: Used to select multiple devices connected to a single controller (such as DECtape units with operator set unit numbers).

BUSY: Indicates that the device is doing internal processing and cannot participate in a data transfer. Need not be used in many devices, READY may be adequate.

ERROR: Indicates the source or cause of an Error Interrupt. Bit 15 is used for single error conditions or may be the logical OR of several error conditions to allow the TST instruction to check error status.

NOTE

DONE and ERROR bits are set by the device to cause an interrupt, and must be cleared by the processor to permit future interrupts. Such clearing may be explicit (by transferring zeros to that location) or implicit (by commanding another operation which automatically clears them).

Table 2-5
Preferred Order of Device Register Assignments

ADDRESS (OCTAL)		
N	CSR	CONTROL STATUS REGISTER
N+2	DBR	DATA BUFFER REGISTER
N+4	MAR	MEMORY ADDRESS REGISTER
N+6	WCR	WORD COUNT REGISTER
N+10	DAR	DEVICE ADDRESS REGISTER

CSR – Device function, status and interrupt control.
DBR – Data register for information transfer.
MAR – Memory location for block transfer. Incremented by device logic after each word transfer.
WCR – Set by program to control length of block transfer.
DAR – Track or block number for mass storage devices.

When several registers are used for the same function, they should be assigned contiguous addresses, and be followed by registers of other functions in the same order as for single registers of each function.

CSR1
CSR2
DBR1
DBR2
DBR3
MAR
WCR
DAR

All register types are optional; only implemented registers should be assigned addresses.

Chapter 3
Interface Circuits and Hardware

3.1 INTRODUCTION

This chapter discusses the specific circuits, modules, and hardware used for interfacing devices to the Unibus.

3.2 CIRCUITS

The Unibus, a high-speed data transmission facility, imposes certain restrictions when attaching other devices to it. The actual bus is a matched and terminated transmission line which must be received and driven with devices designed for that specific application. The following paragraphs describe bus transmission, bus signal levels, bus length, and bus receiver and transmitter circuits.

3.2.1 Unibus Transmission

The actual bus medium consists of several types of cable. The standard cabling is composed of short jumper modules that interconnect the system units within a mounting box. The M920 Module serves as the jumper module. Critical ground signals are also carried on this module. Cables used between BA11 Mounting Boxes consist of a Flexprint® cable assembly with alternating signals and grounds. The characteristics necessary for proper Unibus transmission are:

Characteristic Impedance	120Ω ±15 percent
Resistance	0.135Ω/ft, maximum

Either twisted pair or coaxial cable laid for minimum crosstalk is recommended for long cable lengths and for applications requiring extreme physical durability of the cable.

The Unibus is terminated at each end by a resistive divider for each signal except the grant signals (see Figure 3-1). The grant signals are terminated with a single resistor. Two M930 Terminator Modules are included in every system to provide these functions.

3.2.2 Unibus Signal Levels

The rest state for all Unibus signal lines, except the grant lines BG⟨7:4⟩ and NPG, is a logic 0 of +3.4V. The asserted state (logic 1) is between ground and +0.8V, which is the saturation voltage of the device driving the bus. The rest state for the grant signals is ground (logic 0) and the asserted state (logic 1) is +3.4V. To guarantee operation under worst case conditions, receivers should have a switching threshold of approximately 2V.

Digital Equipment Corporation uses standard terminology to name signal lines, to aid the reader in determining their active state. Either an H or L follows the signal name mnemonic and is separated by a space. This letter indicates the asserted (logic 1) state of the signal to be either high (approximately +3V) or low (ground). Thus, a Unibus data line is called BUS D00 L and a grant line is called BUS BG4 H.

All signals which are not Unibus signals are characterized in terms of standard transistor-transistor logic (TTL) loads. These devices, which are similar to the DEC 7400 Series, have a low state input load of -1.6 mA and a high

®Registered trademark of Sanders Associates.

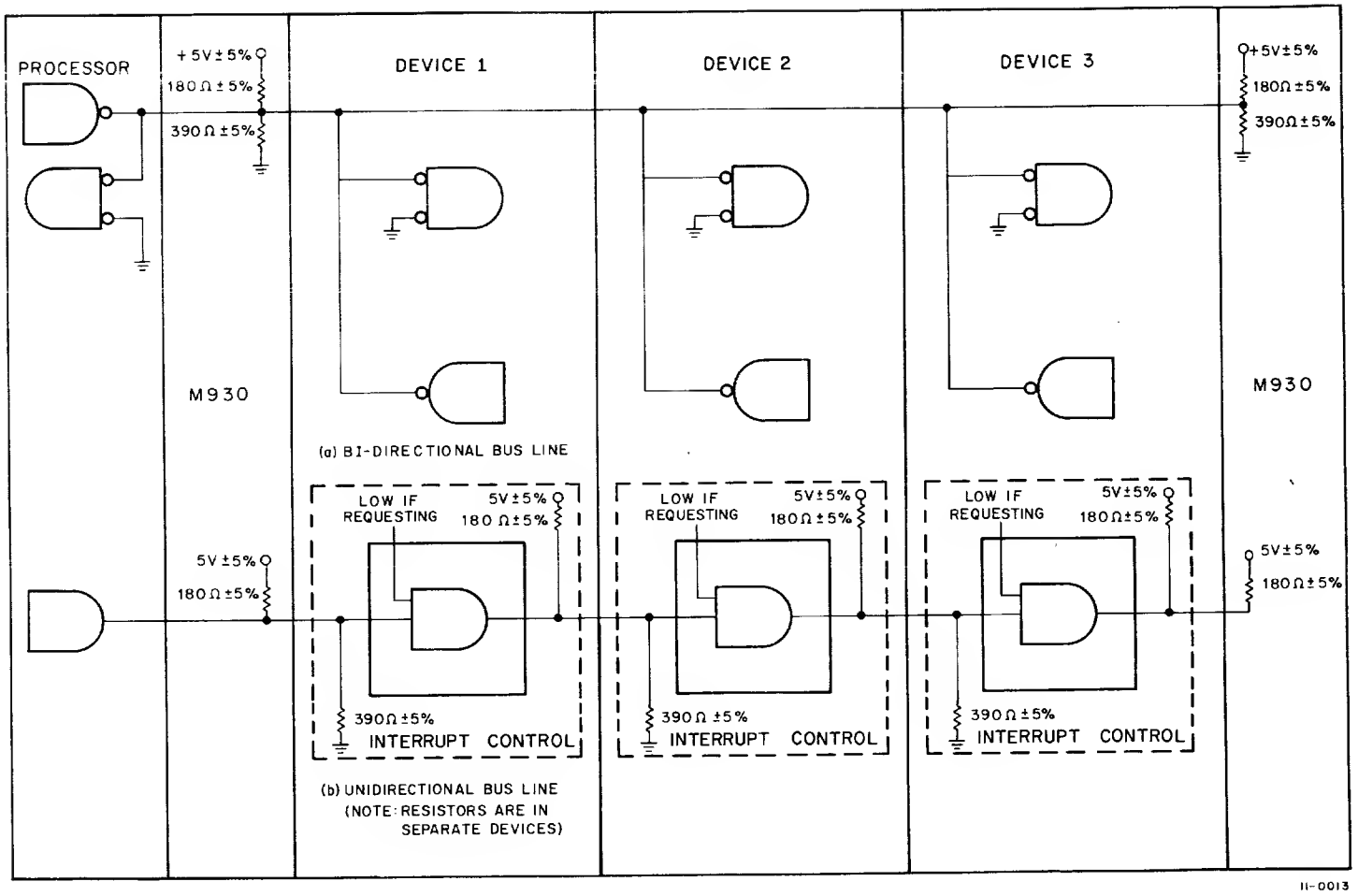


Figure 3-1. Bus Terminations for Bidirectional (a.) and Unidirectional (b.) Bus Lines

state leakage current of 40 μA. Outputs are characterized by the number of inputs they can drive (called fanout). A standard TTL gate (as used in the M113) can drive 10 unit loads.

3.2.3 Unibus Length and Loading

The maximum length of the Unibus is a complex relationship involving the type of cable, the bus loading, and distribution of receiver and transmitter taps on the bus. Since the Unibus is a transmission line, and the transfers are asynchronous and interlocked, the electrical delay imposed by length is not a factor.

With Flexprint cable (Tape Cable S-1680), the maximum reasonable length is 50 feet minus the combined length of all stubs or taps, which are those wires from the actual bus to the receivers and transmitters. This maximum length is obtainable only if the individual tap lengths are less than 18 inches and if the loading is not more than a standard of one receiver and two transmitters. If loads are concentrated at one end of the Unibus and a single load is at a distant point, the maximum length could change, provided that the crosstalk of the employed cable is low enough.

The Unibus is limited to a maximum of 20 unit loads. This limit is imposed because of the loading of receivers and leakage of drivers at the high state. This limit is set to maintain a sufficient noise margin. For more than 20 unit loads, a Unibus repeater option (DB11-A) may be used.

3.2.4 Bus Receiver and Transmitter Circuits

The equivalent circuits of the standard Unibus receivers and transmitters are shown in Figure 3-2. Any device which meets these requirements is acceptable. To perform these functions, Digital Equipment Corporation uses two monolithic integrated circuits with the characteristics listed in Table 3-1. Typical transmitter and receiver circuits are shown in Figure 3-3.

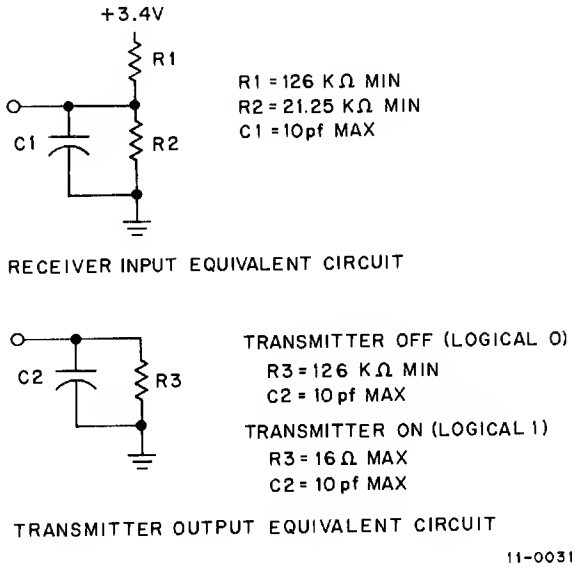


Figure 3-2. Transmitter and Receiver Equivalent Circuits

Table 3-1
DEC Receiver and Transmitter Characteristics

Receiver (DEC 380A)	
Input high threshold	2.5V, maximum
Input low threshold	1.4V, minimum
Input current @ 2.5V	160μA, maximum
Input current @ 0V	25μA, maximum
Output drive	7 TTL unit loads
Transmitter (DEC 8881)	
Input loading	1.25 TTL unit load
Low output voltage @50 mA sink	0.8V, maximum 0.4-0.6V, typical
High output leakage @3.5V	25μA, maximum

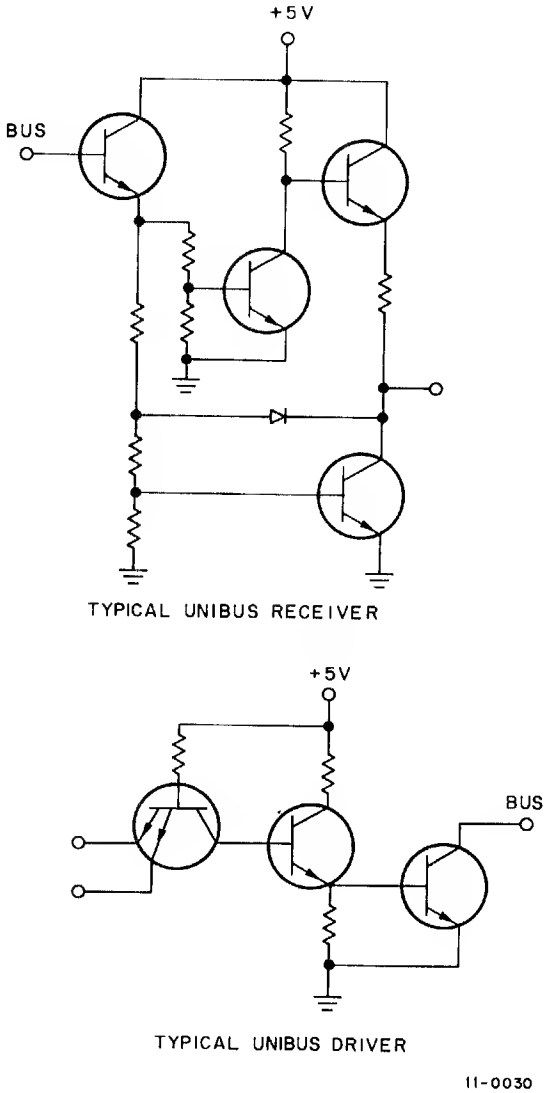


Figure 3-3. Transmitter and Receiver Typical Circuits

3.3 UNIBUS INTERFACE MODULES

The following paragraphs describe modules used for Unibus interfacing. These modules include the jumper module, cable, and transmitter and receiver modules that employ the circuits described in the previous paragraph. Descriptions of the address selector, bus and interrupt control, and general device interface modules are also included.

3.3.1 Unibus Cables

a. Unibus Jumper Module M920 – The M920 Module (see Figure 3-4) is a double module that connects the Unibus from one system unit to the next. The printed circuit cards are on one-inch centers. A single M920 Module carries all 56 Unibus signals and 14 grounds.

b. Unibus Cable BC11A – The BC11A (see Figure 3-5) is a 120-conductor Flexprint cable used to connect system units in different mounting drawers or to connect a peripheral device removed from the mounting drawer.

The 120 signals include all the 56 Unibus lines plus 64 grounds. Signals and grounds alternate to minimize crosstalk. Cable types and lengths are listed below:

Type	Length (ft.)
BC11A-2	2.0
BC11A-5	5.0
BC11A-8F	8.5
BC11A-10	10.0
BC11A-15	15.0
BC11A-25	25.0

3.3.2 Unibus Terminations

The M930 Unibus Terminator Module is a short, double-size module that terminates all signal lines on the Unibus. All pins have a resistive divider termination of 180Ω to +5V and 390Ω to ground, except those listed below:

180Ω to +5V (for grant lines)	Ground Pins		+5-V Input Pins
AV1	AB2	BB2	AA2
AU1	AC2	BC2	
BA1	AN1	BD1	BA2
BB1	AP1	BE1	
BE2	AR1	BT1	
	AS1	BV2	
	AT1		
	AV2		

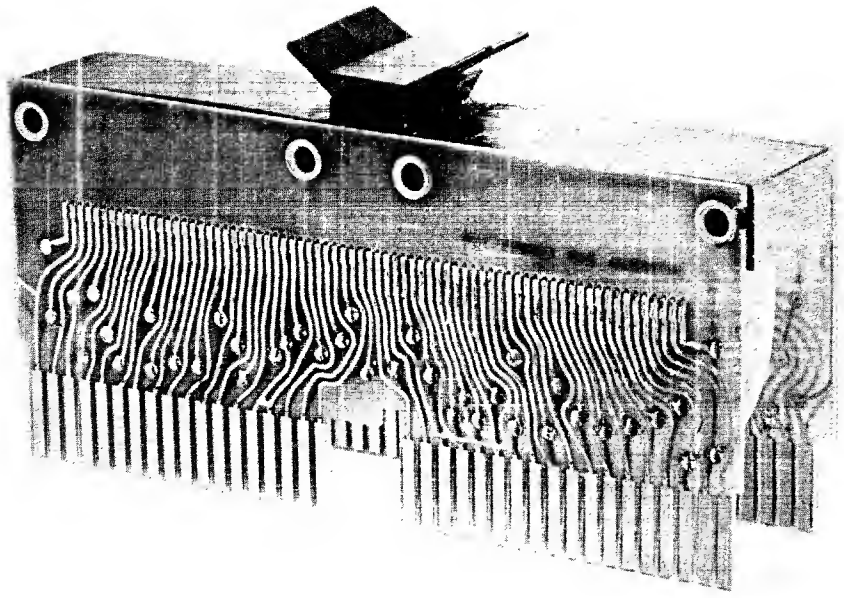


Figure 3-4. Unibus Jumper Module M920

Figure 3-5. Unibus Cable BC11A

3.3.3 Unibus Receivers and Transmitters

a. M783 Unibus Transmitter – This transmitter module contains 12 drivers; 8 drivers have a common gate line, 4 have 2-input positive AND gating. Input loading is 1.25 standard TTL load. The module is a single-height, 8.5 inch-long Flip-Chip. A circuit schematic of the M783 Transmitter is shown in Figure 3-6.

b. M784 Unibus Receiver – This receiver module consists of 16 DEC 380A inverting circuits which receive bus signals and provide a buffered bus signal output. The output fanout is seven standard TTL unit loads. The receiver module is a single-height, 8.5-inch-long Flip-Chip. A circuit schematic of the M784 Receiver Module is shown in Figure 3-7.

c. M785 Unibus Transceiver – This module consists of eight pairs of DEC 8881 Drivers and DEC 380 Receivers which are used for bidirectional interfacing to the Unibus. The drivers and receivers have two common gate lines: one for receivers, one for drivers. The driver input loading is 1.25 standard unit load and the receiver fanout is 7 standard TTL unit loads. The module is a single-height, 8.5-inch-long Flip-Chip. A circuit schematic of the M785 Transceiver Module is shown in Figure 3-8.

3.3.4 M105 Address Selector Module

The M105 Address Selector Module provides gating signals for up to 4 full 16-bit device registers. A block diagram of this module is shown in Figure 3-9. Note that IN and OUT are always used with respect to the master (controlling) device. Thus, when the M105 is used in a peripheral device, an OUT transfer is a transfer of data out of the master (such as the central processor) and into the device. Likewise, an IN transfer is the operation of the peripheral furnishing data to the central processor. The M105 Module is described more fully in following paragraphs.

3.3.4.1 **Inputs** – The M105 Module input signals consist of 18 address lines, A⟨17:00⟩; 2 bus control lines, ⟨1:0⟩; and a master synchronization ⟨MSYN⟩ line. The address selector decodes the 18-bit address on lines A⟨17:00⟩ as described below. This address format, used for selecting a device register, is shown in Figure 3-10. Note that all inputs are standard bus receivers.

a. Line A00 is used for byte control.

b. Lines A01 and A02 are decoded to select one of the four addressable device registers.

c. Decoding of lines A⟨12:03⟩ is determined by jumpers on the module. When a given line contains a jumper, the address selector searches for a zero on that line. If there is no jumper, the address selector searches for a one.

d. Address lines A⟨17:13⟩ must be all ones. This specifies an address within the top 8K byte address bounds for device registers.

3.3.4.2 Slave Sync (SSYN) – When SSYN INH is grounded, it inhibits the acknowledgment signal (SSYN) normally generated by the M105. In this case, the SSYN must be generated by another source. When SSYN INH is not grounded, SSYN is returned to the master 100 ns after register select becomes true. This time may be extended to a maximum of 400 ns by adding an external capacitor between SSYN INH and ground.

3.3.4.3 **Outputs** – The M105 Output Signals permit selection of four 16-bit registers and provide three signals used for gating information to and out of the master device. The M105 may be used instead to select up to eight 8-bit registers, or any appropriate combination of byte and word registers.

The input signals select the M105 control output line states as shown in Tables 3-2 and 3-3.

3.3.4.4 **Specifications** – The M105 output fanout is ten standard TTL loads for register select lines and eight standard TTL loads for gating control lines. The module is a single-height, 8.5-inch-long Flip-Chip. A circuit schematic for this module is shown in Figure 3-11.

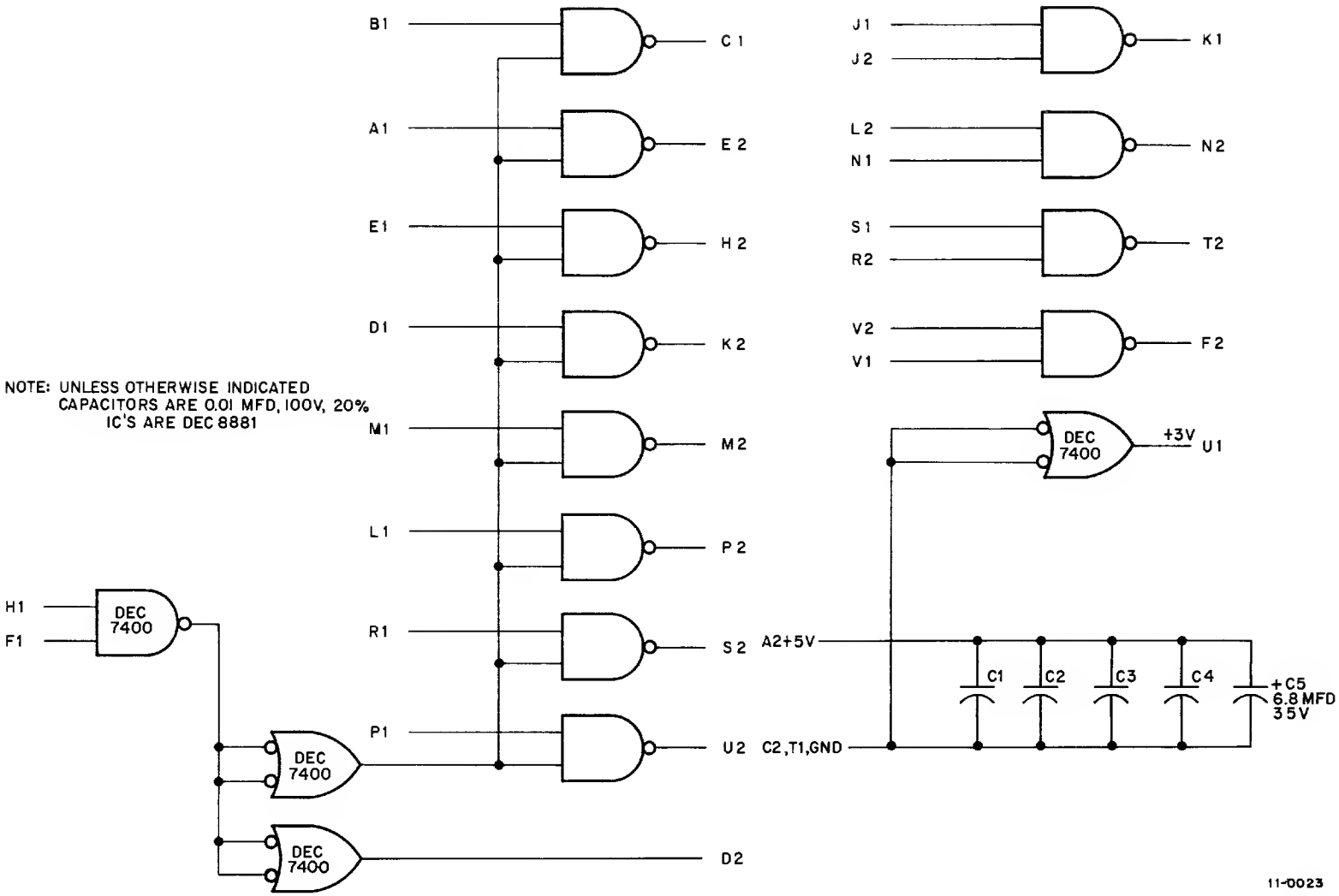


Figure 3-6. M783 Unibus Transmitter (schematic diagram)

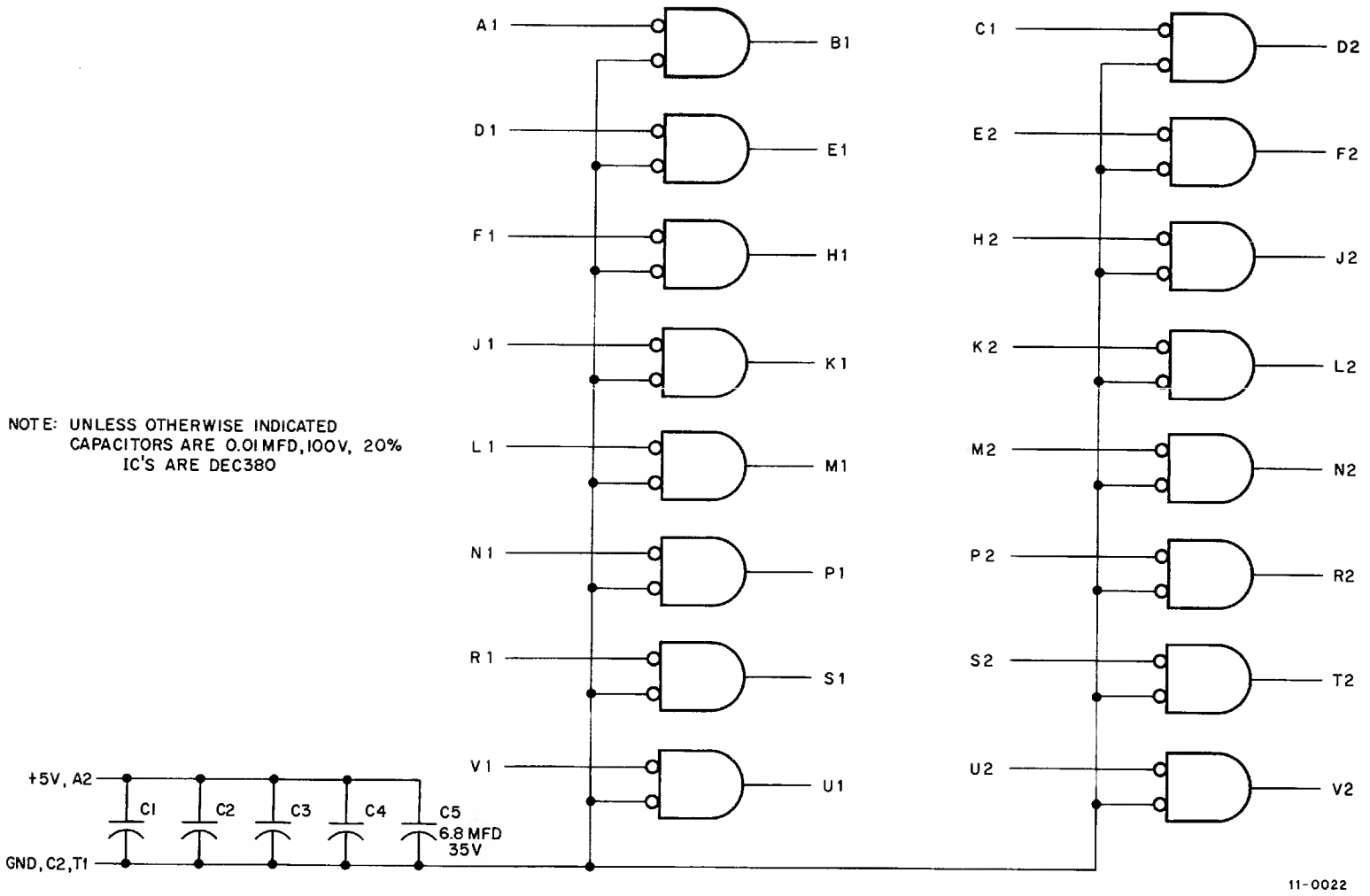


Figure 3-7. M784 Unibus Receiver (schematic diagram)

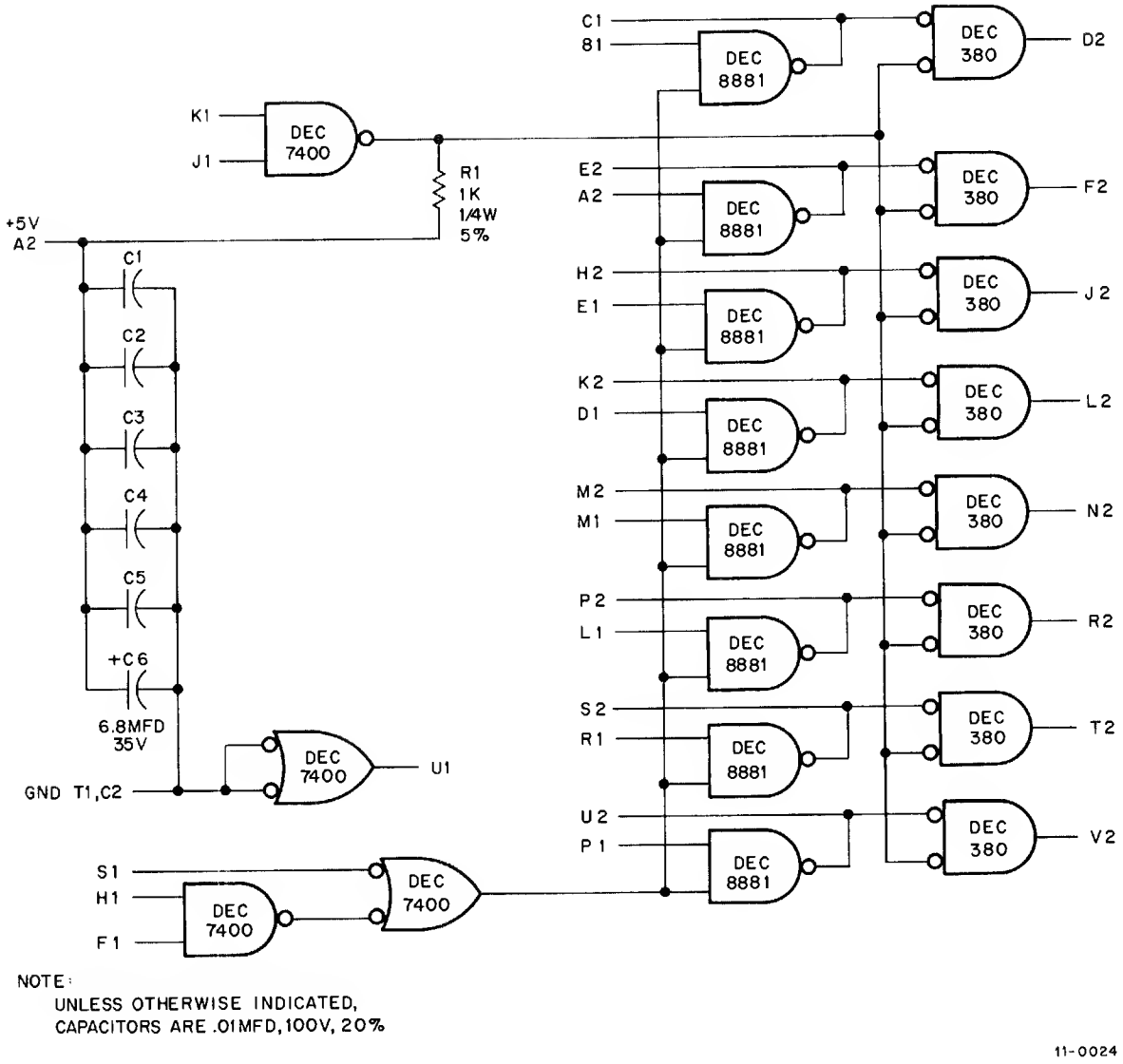


Figure 3-8. M785 Unibus Transceiver (schematic diagram)

Table 3-2
M105 Select Lines

Input Lines A⟨02:01⟩	Select Lines True (+3V)
00	0
01	2
10	4
11	6

NOTE

1. Lines A⟨17:13⟩ must be all 1s (0V on Unibus).
2. Lines A⟨12:03⟩ are selected by jumpers.

Table 3-3
Gating Control Signals

Mode Control C⟨1:0⟩	Byte Control A00	Gating Control Signals True (+3V)	Bus Sequence
00	0	IN	DATI
00	1	IN	DATI
01	0	IN	DATIP
01	1	IN	DATIP
10	0	OUT LOW OUT HIGH	DATO
10	1	OUT LOW OUT HIGH	DATO
11	0	OUT LOW	DATOB
11	1	OUT HIGH	DATOB

NOTE
Gating control signals may become true although select lines are not.

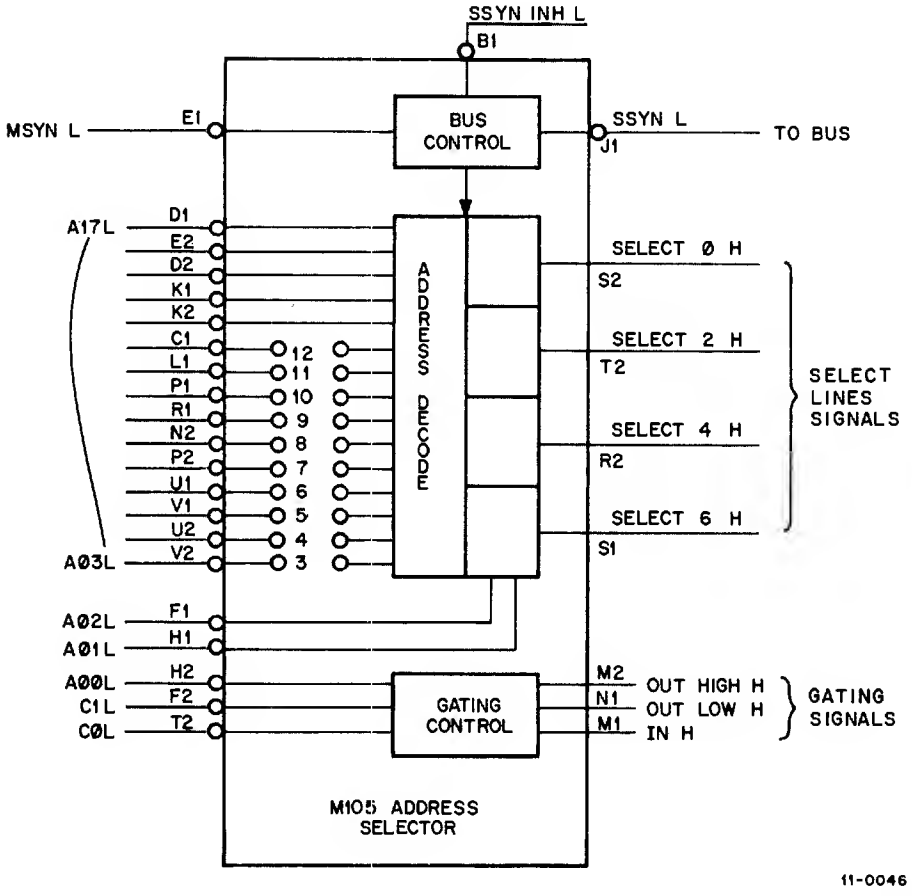


Figure 3-9. M105 Address Selector

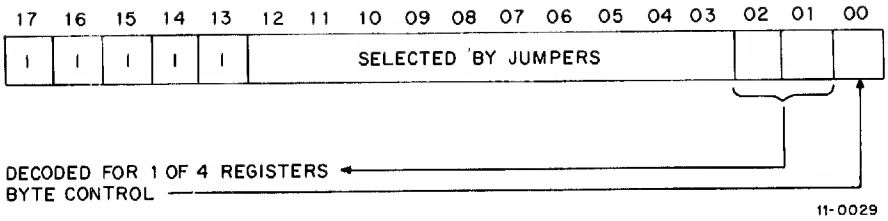


Figure 3-10. Device Register Select Address Format

3.3.5 M782 Bus and Interrupt Control Module

The M782 Bus and Interrupt Control Module provides the circuits and logic required to make bus requests and to gain control of the bus (become bus master). The module also includes circuits needed to generate an interrupt, if desired. The module contains two completely independent request and grant acknowledge circuits (channels A and B) for establishing bus control. The interrupt control circuit can be used with either, or both, of the request channels and provides a unique vector address for each channel. Figure 3-12 is a block diagram of the M782 Module, which is a single-height, 8.5-inch-wide Flip-Chip.

The master control section (either channel A or B) is used to gain control of the bus. When the INTR and INTR ENB requesting inputs are asserted, a bus request is made on the BR level corresponding to the level of the BR line wired to the BR pin of the module. When the priority arbitration logic in the system recognizes the request and issues a bus grant signal, the master control circuit acknowledges with a SACK signal. When the device has fulfilled all requirements to become bus master, the master control circuit asserts BBSY and then asserts a MASTER signal. (Refer to Section 2.4.1.)

Once the device has gained bus control by means of a BR request, an interrupt can be generated. If an interrupt is desired, the module is interconnected as shown in Figure 3-13. This figure illustrates the use of the two channels to first generate requests for bus control and then initiate interrupts. The request from channel A is a slightly higher priority than the channel B request because the bus grant signal first enters A, then enters B.

The vector address is selected by jumpers on the M782 Module. Since the vector is a two-word (four-byte) block, it is not necessary to determine the state of bits 0 and 1. The six selectable lines determine the vector address. The least significant line is controlled by the VECTOR BIT 2 input signal. If this input is asserted, then bus line D02 is asserted. Thus, the interrupt on channel A uses a vector at location 100 and channel B uses a vector at location 104.

Figure 3-14 illustrates an M782 Module used for bus control in a device that directly transfers data to memory and then causes an interrupt when the transfer is completed. Channel A is connected to the NPR and NPG lines and is used to gain bus control for direct to memory, or device-to-device, transfers. Channel B is used to gain bus control for an interrupt.

The circuit schematic for the M782 Module is shown in Figure 3-15. Each master control section contains two flip-flops that sequence through four states, thereby controlling the request for bus control. Figure 3-16 is a state diagram of this sequence. The BG IN signal is allowed to pass through the module to BG OUT when the device is not issuing a request (state A), is master (state D), or has had the request honored (state E). To request bus use, the AND condition of INTR and INTR ENB must be satisfied. These levels must be true at least until the request is granted. Once bus control has been attained, it can be released by either asserting CLEAR or by negating

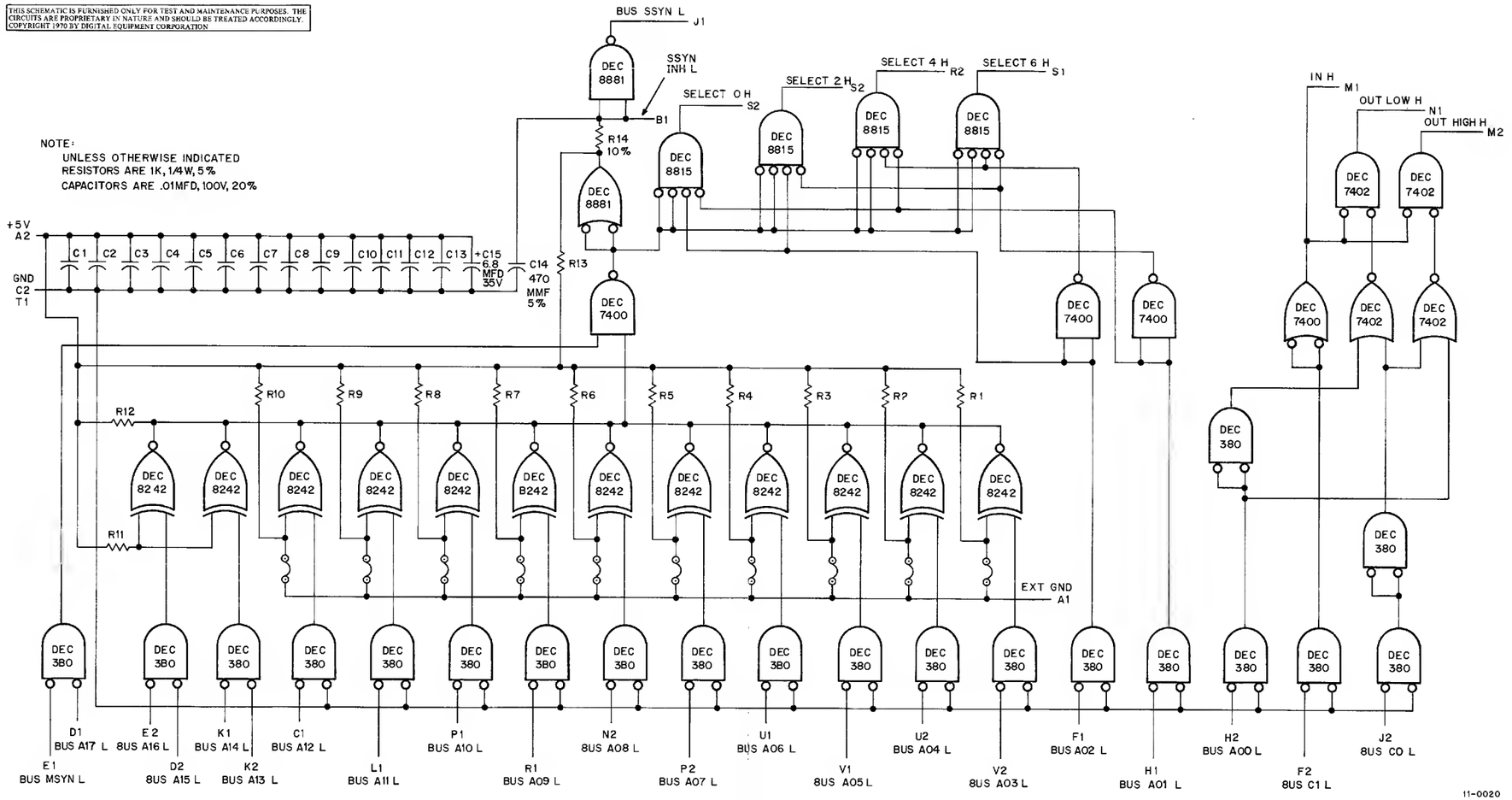


Figure 3-11. M105 Address Selector (schematic diagram)

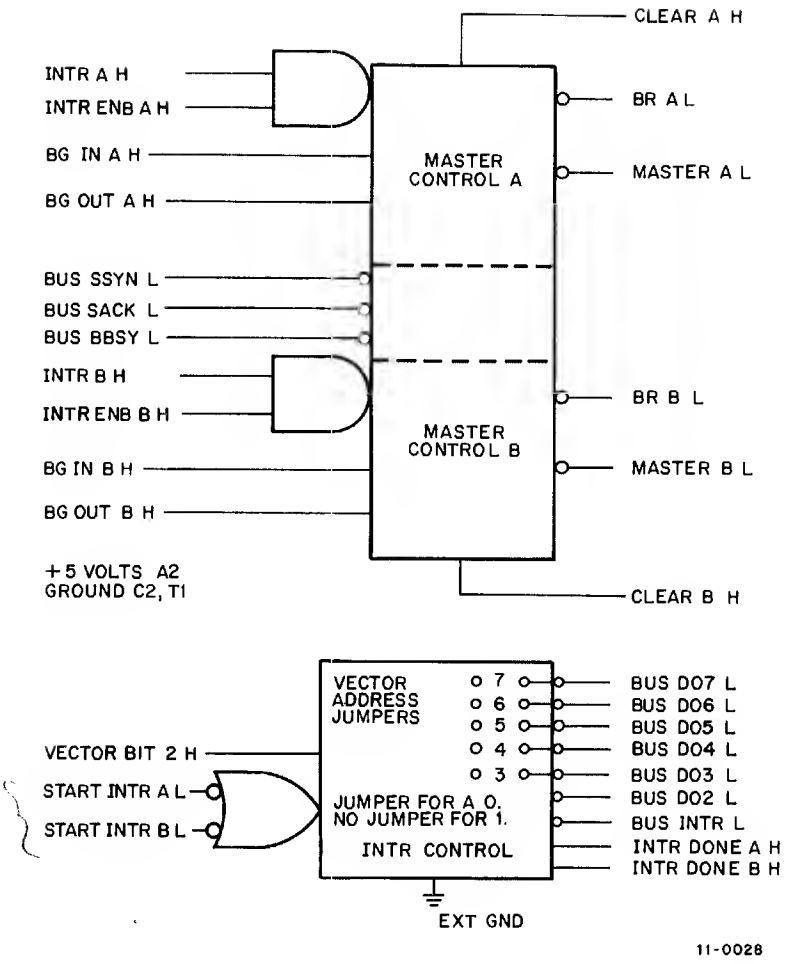


Figure 3-12. M782 Interrupt Control (block diagram)

either INTR or INTR ENB. The first method leaves the master control in state E, thereby inhibiting further bus requests even if INTR and INTR ENB remain asserted. This prevents multiple interrupts when the master control is used to generate interrupts. The second method is used to release the bus after NPR use.

A summary of all M782 signals is listed in Table 3-4.

3.3.6 DR11-A General Device Interface

The DR11-A Device Interface is a three-module set that plugs into either a small peripheral slot in the KA11 Processor or into one of four slots in a DD11 Small Peripheral Mounting Panel. The DR11-A provides the logic and buffer register necessary for transfers of 16-bit input and output data between the PDP-11 System and an external device.

The DR11-A contains three functional sections: a 16-bit buffer register, a 16-bit data input circuit, and a 2-channel flag and interrupt control. These functional elements are shown in Figure 3-17. Address and bit assignments are shown in Figure 3-18. The DR11-A contains three physical modules: an M105 Address Selector, an M782 Interrupt Control, and an M786 General Register.

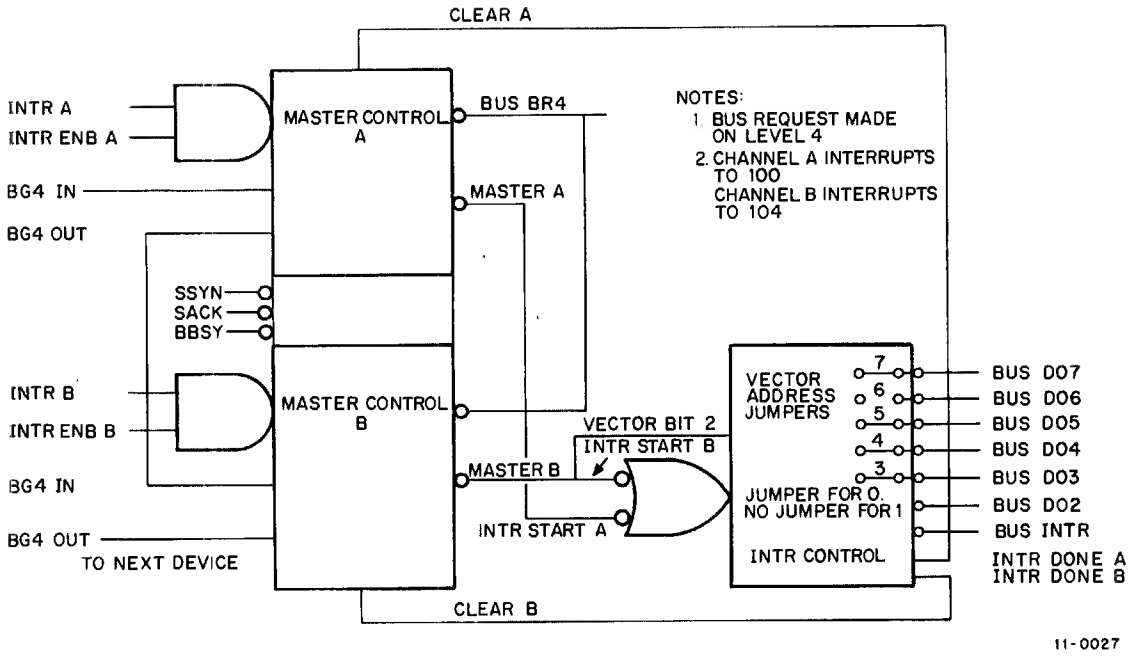


Figure 3-13. M782 Interconnection for 2-Channel Interrupt

Table 3-4
Summary of M782 Signals

Signal	Assertion Level	Input Loading	Output Drive
INTR A, B	H	1 TTL (each)	
INTR ENB A, B	H	1 TTL	
CLEAR A, B	H	1 TTL	
MASTER A, B	L		10 TTL
START INTR A, B	L	2 TTL	
INTR DONE A, B	H		10 TTL
BG IN A, B	H	1 R*	
BG OUT A, B	H		2 D**
BR A, B	L		1 D
VECTOR BIT 2	H	1 TTL	
BUS SSYN	L	1 R	
BUS BBSY	L	1 R	2 D
BUS SACK	L		2 D
BUS INTR	L		1D
BUS D⟨07:02⟩	L		1D

*R = Standard Unibus receiver load.
**D = Standard Unibus transmitter (driver) output.

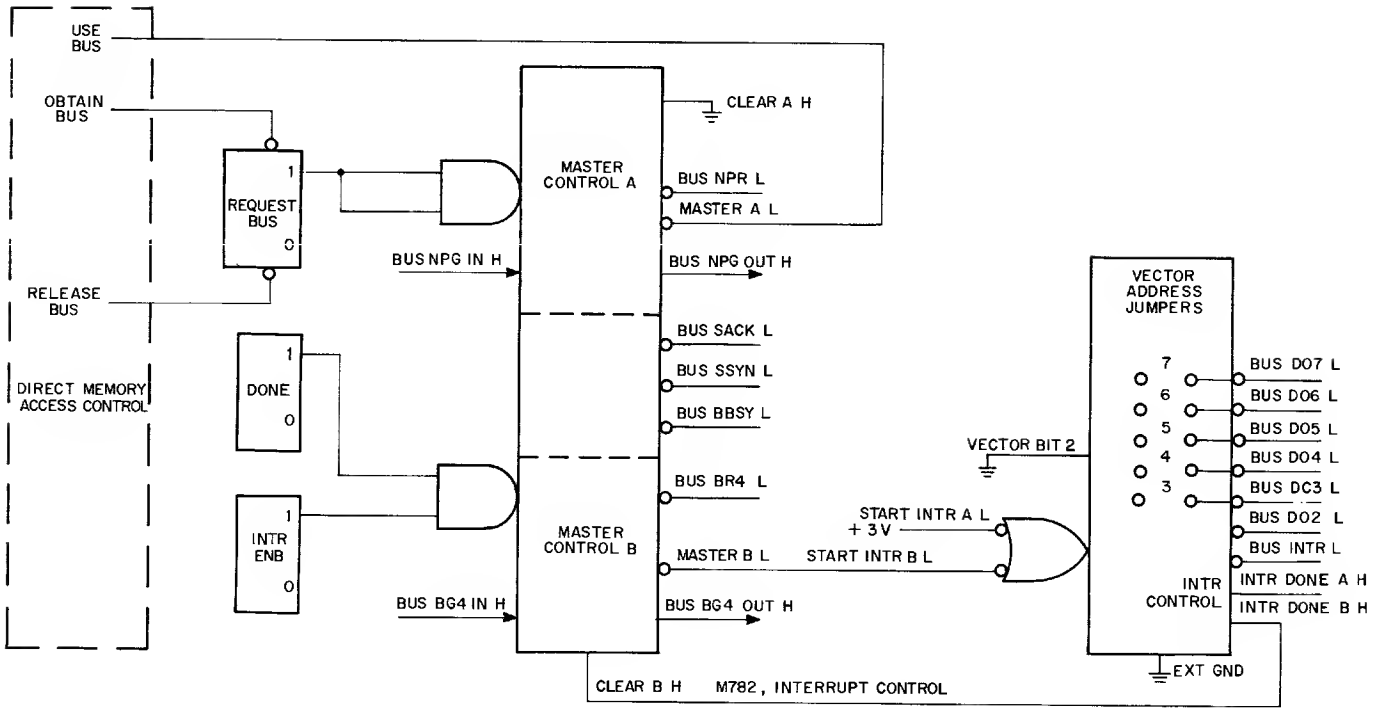


Figure 3-14. M782 Interconnection for Direct Memory Access

NOTE: UNLESS OTHERWISE INDICATED
RESISTORS ARE 1K, 1/4 W, 5%
CAPACITORS ARE 0.01 MFD, 100V, 20%
TRANSISTORS ARE DEC 30098
⌀ = SPLIT LUGS
---- = JUMPERS

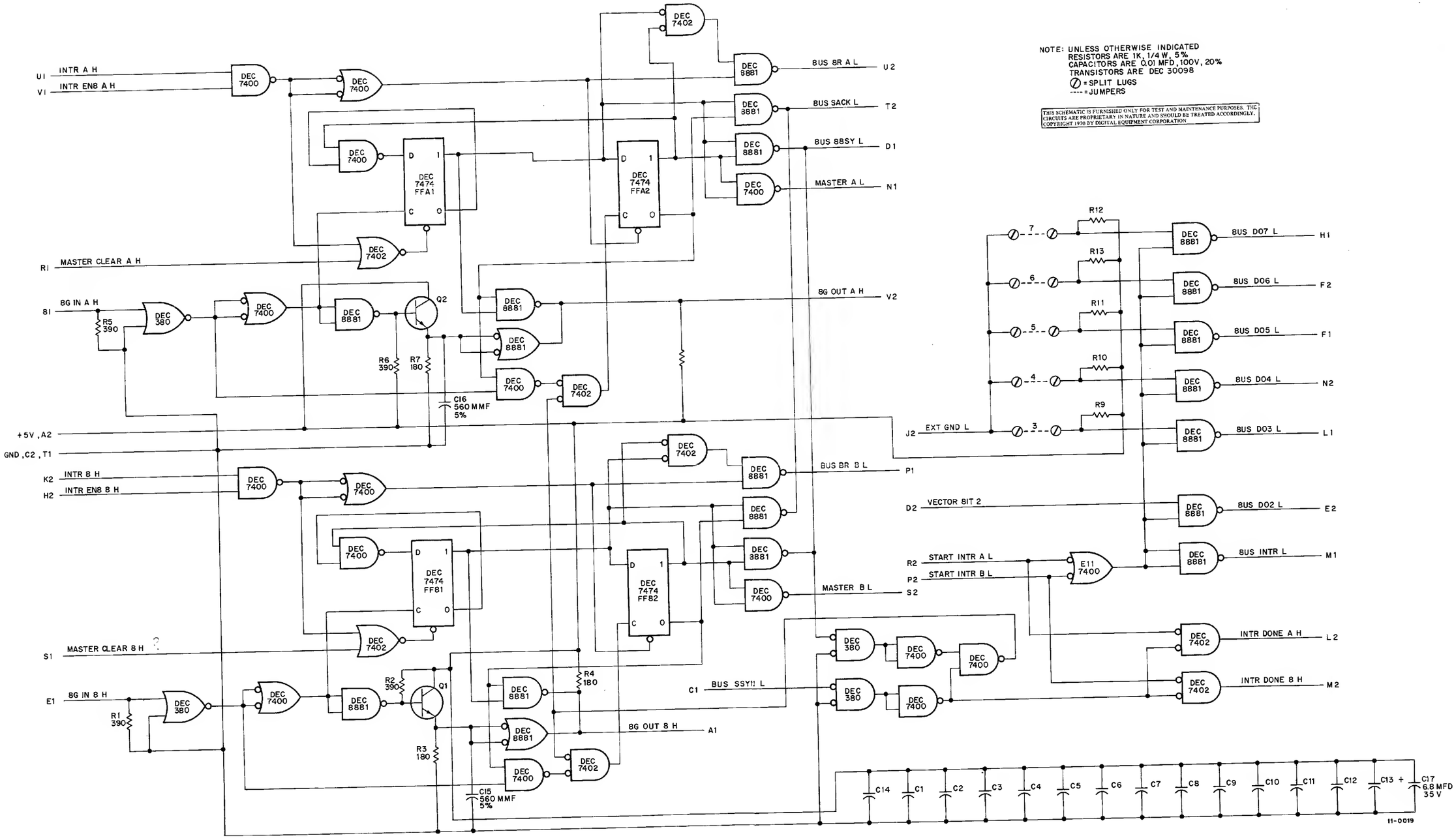
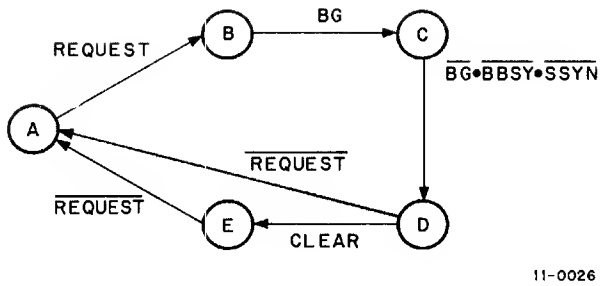


Figure 3-15. M782 Interrupt Control (schematic diagram)



State	FFA1	FFA2		Signals Asserted
A	0	0	Not requesting	None
B	0	0	Requesting	BR
C	1	0	Granted	SACK
D	1	1	Master	BBSY, MASTER
E	0	1	Request honored	None

Figure 3-16. State Diagram of Master Control

The 16-bit buffer register is an addressable register that may be read or loaded by instructions transmitted through the Unibus (see Figure 3-18). The register outputs, together with a control signal (NEW DATA READY) used to indicate that the register has been loaded from the Unibus, are available on a printed circuit edge connector which is mounted on the M786 Module. These signals are logic levels of either +3V (true) or 0V (false). The connector accepts an M927 Cable Connector, which contains solder lugs and can be used with ribbon cable, twisted pair cable, or open wire. The M927 is electrically identical to the M904 Module described in the Logic Handbook.

The interface input circuits consist of 16 bus drivers gated to the bus when the input register is read by a DATI bus sequence (see Figure 3-18). The 16 input lines are +3V if true or OV if false. These signals are also applied to the M786 Module through an M927 Cable Connector and a second printed circuit connector. When a DATI sequence occurs, a pulse signal (DATA TRANSMITTED) is applied to the external device.

Two additional request lines are furnished and may be asserted (+3V) by the external device to initiate an interrupt or to generate a flag that may be tested by the program. Whether these two request lines cause an interrupt is determined by two interrupt enable flip-flops which form part of the status and control register in the option (see Figure 3-18). The request lines form two more bits of the status register, independent of the status of the enable flip-flops; thus, they may be tested by the program.

The priority level of both interrupts must be the same, with interrupt A on a higher sublevel than interrupt B. The M786 contains a priority jumper plug which is normally set at BR5. The interrupt enable flip-flops are cleared to zero (inhibit interrupt) by the occurrence of an INIT signal on the Unibus, or may be set or cleared by the program.

The DR11-A pin assignments are listed in Table 3-5. All inputs are one standard TTL unit load. Inputs have diode protection clamps to ground and +5V. All signals are +3V if true. All outputs are TTL levels capable of eight unit loads. The new data ready and data transmitted signals are positive pulses, approximately 2 μs in duration.

Figure 3-19 is a circuit schematic of the M786 Module. An application of the DR11-A Option is discussed in Section 4.

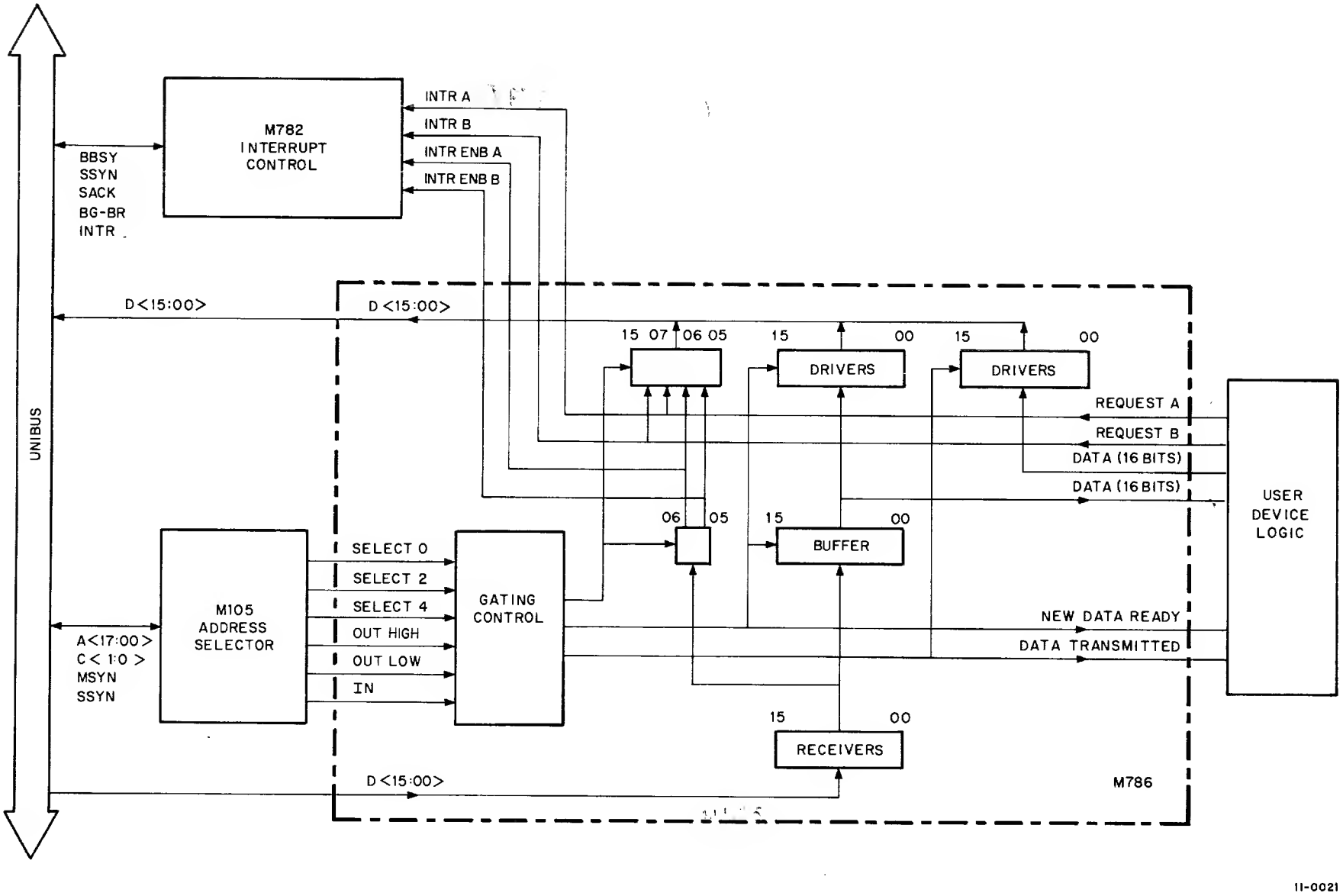


Figure 3-17. DR11-A General Device Interface (block diagram)

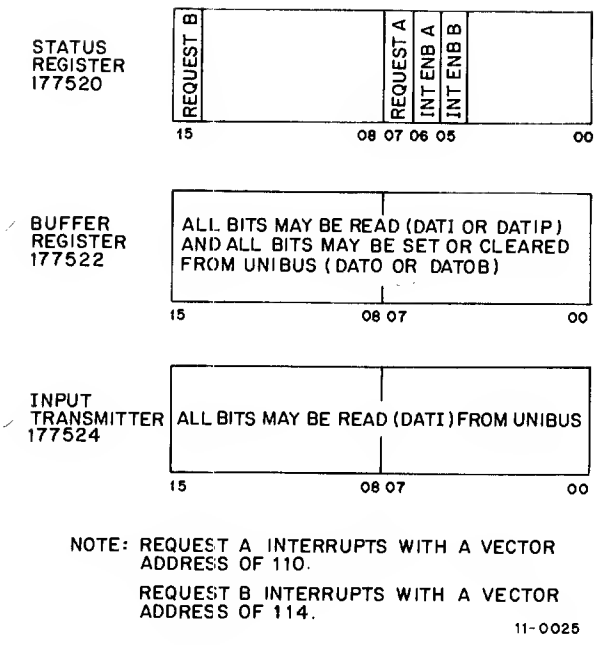


Figure 3-18. DR11-A Address and Bit Assignments

Table 3-5
DR11-A Pin Assignments

Inputs			Outputs		
Signal	Connector	Pin	Signal	Connector	Pin
IN00	2	S1	OUT00	1	P2
IN01	2	S2	OUT01	1	M2
IN02	2	P1	OUT02	1	S2
IN03	2	L1	OUT03	1	P1
IN04	2	P2	OUT04	1	K2
IN05	2	K2	OUT05	1	M1
IN06	2	M1	OUT06	1	S1
IN07	2	T2	OUT07	1	L1
IN08	2	M2	OUT08	1	J1
IN09	2	D2	OUT09	1	H2
IN10	2	E1	OUT10	1	E2
IN11	2	D1	OUT11	1	H1
IN12	2	H1	OUT12	1	D2
IN13	2	E2	OUT13	1	EI
IN14	2	B1	OUT14	1	D1
IN15	2	J1	OUT15	1	BI
REQUEST A	1	T2	NEW DATA RDY	1	V2
REQUEST B	2	H2	DATA TRANSMITTED	2	V2

3.4 PDP-11 INTERFACE HARDWARE

The basic design of the PDP-11 System permits highly flexible and easily varied system configurations because of the modular construction. These advantages are available to the user when designing and building custom interface equipment.

The basic building block of the PDP-11 System is the system unit. The system unit is a die-cast zinc frame. Three 288-pin connector blocks are fastened to this frame and each block contains slots for mounting 8 single-size Flip-Chip Modules. Peripheral devices and controllers may be made up from one or more of these assemblies. Up to six such system units can be mounted in a BA11 Mounting Box. In addition to space for the system units, the mounting box has space for an H720 Power Supply.

The Unibus is interconnected from system unit to system unit within a single mounting box by M920 Unibus Connectors. A BC11-A Cable Assembly carries the Unibus between mounting boxes.

The basic PDP-11/20 System (see Figure 3-20) occupies four of the six positions within the mounting box. Three of these are system units for the KA11 Central Processor, including space for two small peripheral controllers, and one is a system unit for the MM11-E Core Memory. System units mounted in the BA11 Mounting Box can accept either the extended-length Flip-Chip modules (8.5 inches long) or standard-length Flip-Chip modules (5 inches long) if equipped with H850 Module Handle Extenders.

Users who prefer to mount the interface equipment directly into a rack can use H911 Mounting Panels and the BC11A Cable Assembly to extend the Unibus. Information on the H911 Panels as well as the M-Series Logic Modules is provided in the DEC 1970 Logic Handbook.

The following paragraphs provide a detailed description of available hardware specifically designed for the PDP-11 System as an aid in interfacing and building custom equipment.

3.4.1 BB11 Blank Mounting Panel

The BB11 Blank Mounting Panel (see Figure 3-21) is a prewired system unit designed for general interfacing. The unit contains three 288-pin blocks assembled end-to-end in a casting which can be mounted in either the basic PDP-11 Mounting Box or the extension box. Bus and power connectors, described below, use only 6 of the module slots, thereby leaving 18 slots available for customer use.

The BB11 is wired to accept the Unibus in slots A1 and B1 as shown in Figure 3-22. This connection can be made with an M920 Unibus Connector or a BC11A Unibus Cable Assembly. All bus signals, including grant signals, are wired directly to corresponding pins in slots A4 and B4. From this point, the Unibus can be continued to the next unit by using an M920 or BC11A. If the BB11 is the last unit on the bus, slot A4-B4 accepts the M930 Bus Terminator Module. Standard bus pins are described in Appendix C.

Slot A3 accepts the G772 Power Connector (furnished as part of the BA11 Mounting Box). Power of +5V is distributed to all A2 pins; -15V is distributed to all B2 pins except in slots A1, B1, A4, and B4; and ground is maintained through the frame and power connector on pins C2 and T1 of all slots.

3.4.2 BA11 Mounting Boxes

The BA11 Mounting Box (see Figure 3-23) is designed to house the system units that make up a PDP-11 System. The box also includes space for mounting the H720 System Power Supply. The mounting box contains fans for forced air cooling, an insulated top cover (not shown) to prevent debris from falling into the wire-wrap pins (this is necessary because the system units are mounted with the pins up and the modules down), and a foam-lined

NOTES:
UNLESS OTHERWISE INDICATED:
DIODES ARE D664
RESISTORS ARE 390Ω, 1/2W, ± 5%
CAPACITORS ARE 0.01 μF, 100V, 20%

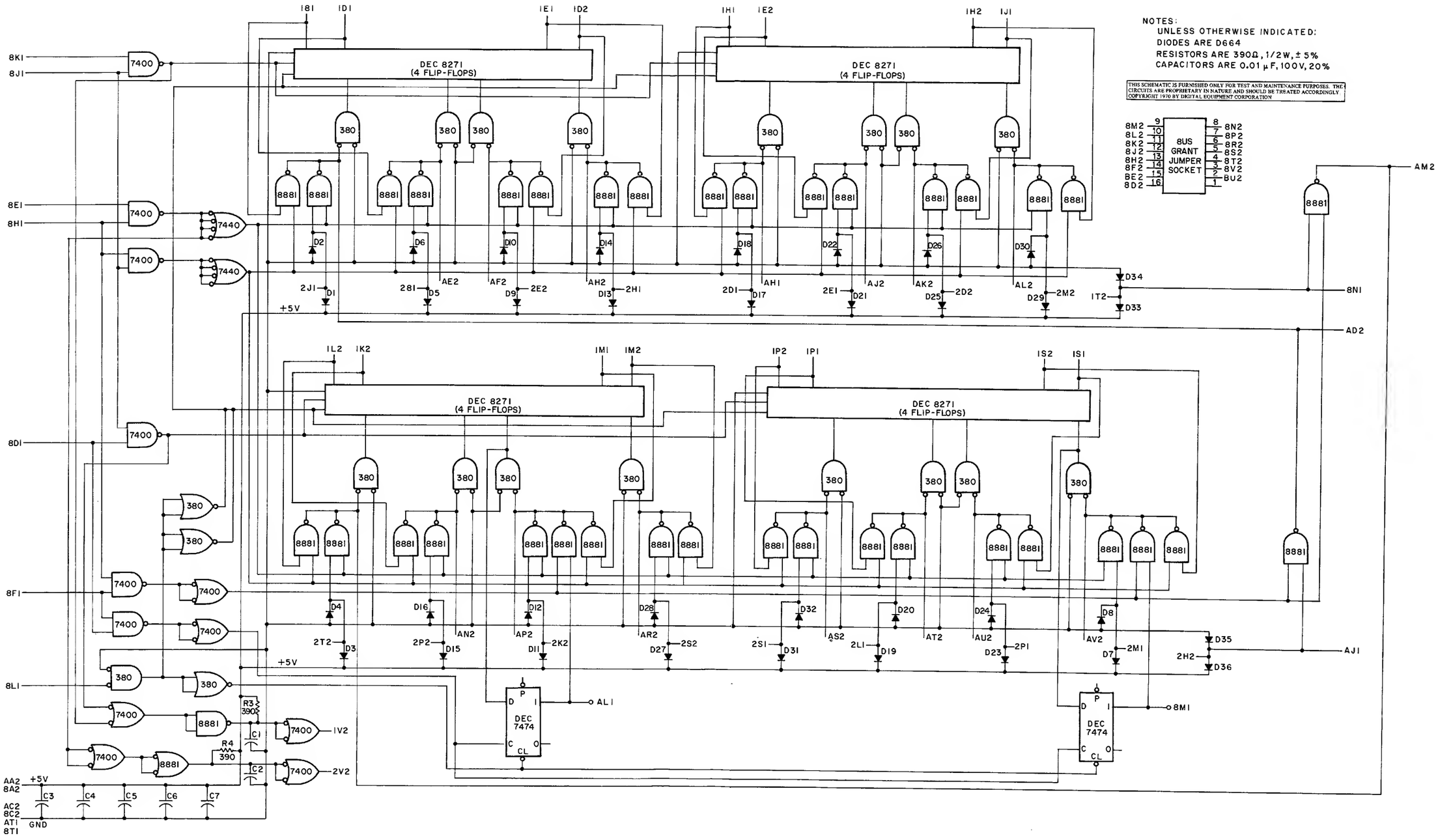


Figure 3-19. M786 Circuit Schematic

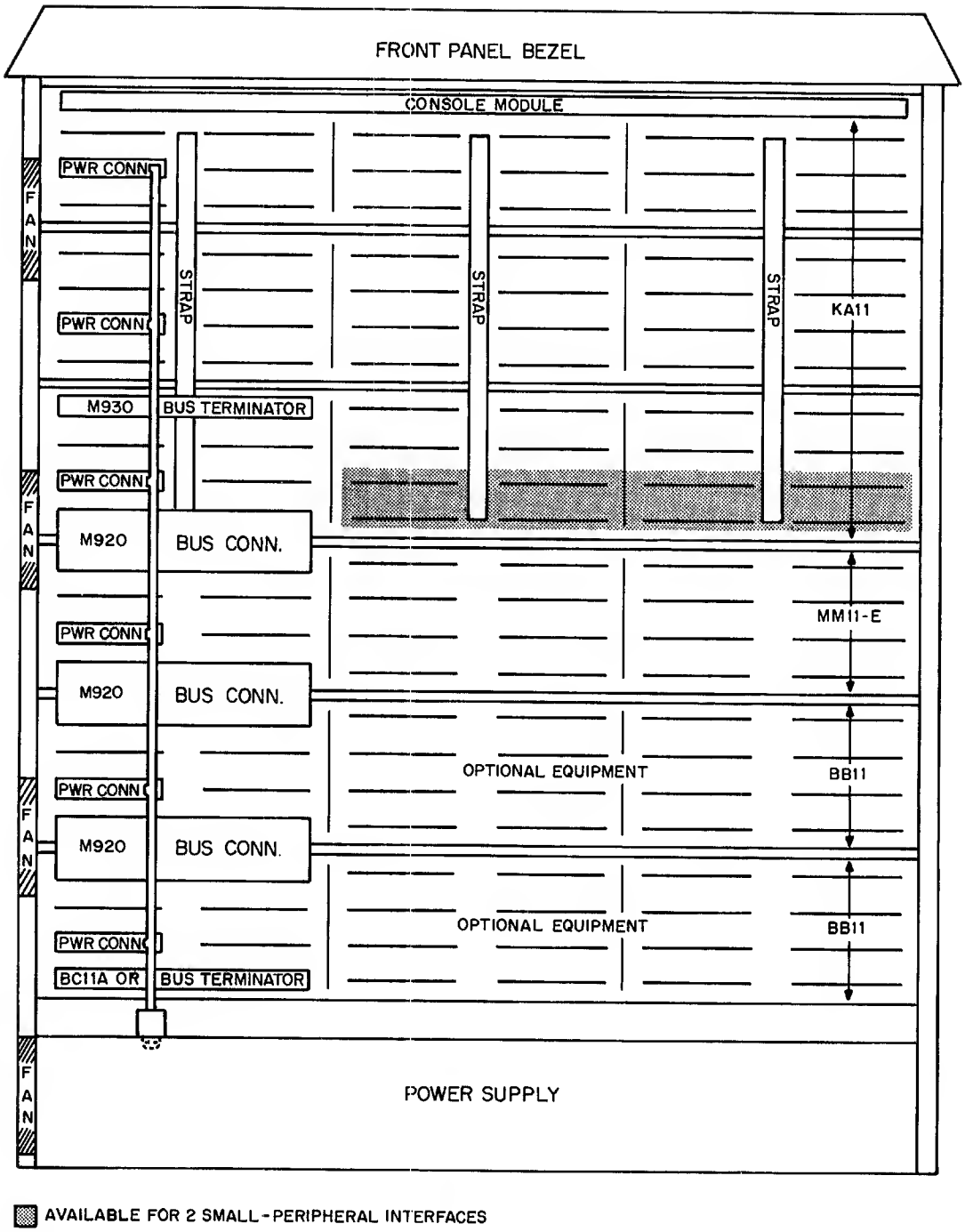


Figure 3-20. Basic PDP-11/20 Layout with Two BB11 Panels (from module side)

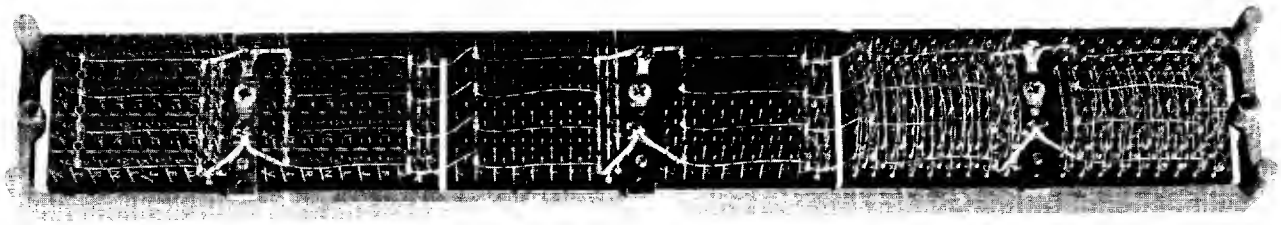


Figure 3-21. BB11 System Unit

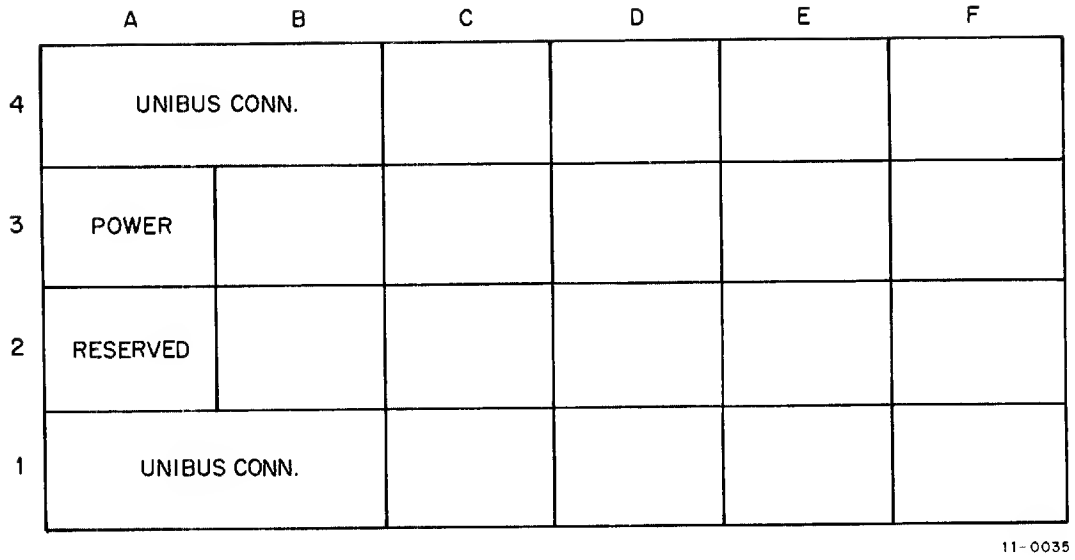


Figure 3-22. BB11 Module Layout

bottom cover, which serves as a module retainer and minimizes module vibration. The bottom cover serves as part of the air plenum to ensure adequate cooling.

The mounting box is fabricated from zinc-plated steel to resist corrosion. Detailed mounting information, including dimensions, is included in the Installation Planning Section of the PDP-11 Handbook.

The four available models are listed in Table 3-6.

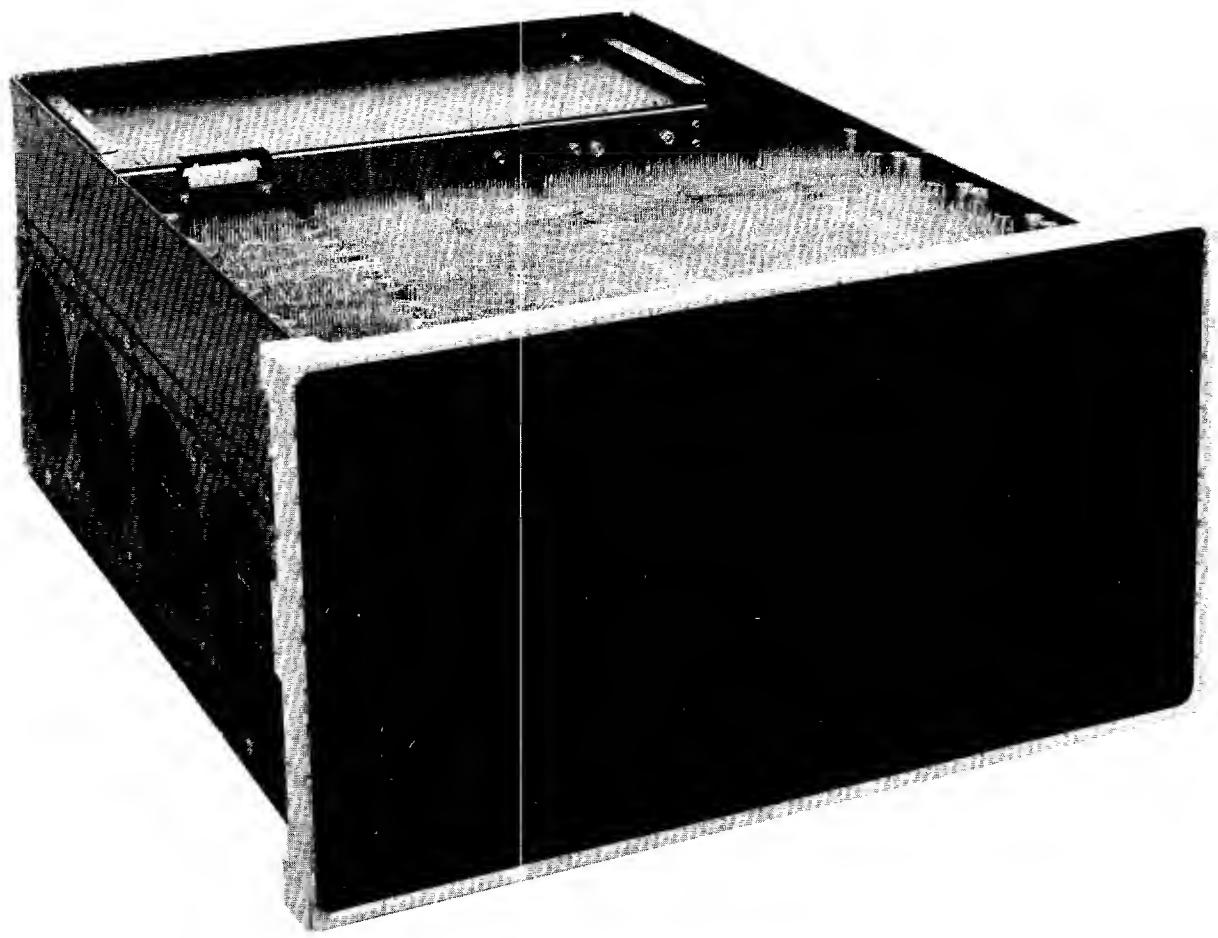


Figure 3-23. BA11 Mounting Box (with wired System Units and H720 power Supply installed)

Table 3-6
BA11 Mounting Box Models

Model No.	Description
BA11-CS	Basic box*, with tilt-slides for rack mounting
BA11-CC	Basic box, with table-top decorative cabinet
BA11-ES	Extension box**, with tilt-slides
BA11-EC	Extension box, with table-top cabinet

* The basic box includes proper bezel for the console and a key-operated power switch.

** The extension box includes blank front with decorative bezel and does not include a power switch.

3.4.3 H720 Power Supply

The H720 Power Supply is specifically designed to provide power for PDP-11 Systems. It provides +5V and -15V regulated power as well as additional power for console functions. The +5V portion is protected by overvoltage circuits and both portions of the supply feature dynamic current limiting. The basic regulating element is a high efficiency switching regulator that keeps voltages within a ±3 percent tolerance. In addition to providing power, the H720 Supply generates three other signals: a line frequency signal that approximates a squarewave referred to as LTC (line time clock), and the ACLO and DCLO signals previously described in Paragraph 2.2.3. The low voltage detection circuits are interlocked so that -15V is not established until +5V is established. If the +5V fails, the -15V section is shorted to ground to prevent damage to system logic circuits. The power supply outputs are listed in Table 3-7.

There are two H720 Power Supply Models. The H720-A is designed for use with an input of 120V, ±10 percent. The H720-B is designed for an input of 240V, ±10 percent. In addition, the H720-B has taps for lower voltages of 232, 225, and 217, ±10 percent. Both models are designed to operate within a 47- to 63-Hz frequency range.

Table 3-7
H720 Power Supply Outputs

Output	Regulation	Capacity	Remarks
+5V	±3%	14A	
-15V	±3%	10A	
+8V, rms	±15%	1.5A	Full-wave, unfiltered
-24V	±20%	1.0A	Unregulated, filtered
LTC L	0 to 5V		Line time clock
ACLO L	Logic level		ac line low
DCLO L	Logic level		dc line low

Chapter 4
Interface Examples

4.1 INTRODUCTION

Examples of interface designs in Paragraphs 4.2 to 4.10 use the techniques and equipment described in previous chapters. To draw attention to the design features of each interface type, a series of related examples is presented. The first example is a simple basic interface. Each additional example implements several features by adding logic circuits to the previous example. Thus, the first example is the simplest possible read/write interface. This circuit is then used with additional logic to form a program-controlled interface, which in turn is used with additional circuits to form an interrupt-serviced interface, until finally, the circuit is used with additional circuits to form a direct-memory-access interface.

The examples cover input and output transfers and also illustrate techniques for combining the two functions into one interface. Each example includes a description of the operation and logic of the interface, a typical implementation, and programming methods that might be used to operate a device with the interface.

4.2 BASIC INTERFACE

The simplest possible interface, a basic read/write interface, is used when data is transferred to and from the register during bus operations. Applications of read/write, read-only, and write-only registers are discussed in Paragraph 2.7. This particular read/write interface consists of only a storage register and bus gating circuits. The register may be used either as a data register or may be used to drive an output device, such as a set of indicator lights.

4.2.1 Interface Operation

When the basic read/write interface is used, data transfers are under control of the program and the register is assigned an address on the Unibus. During execution of an instruction that addresses the interface, the processor conducts a bus data transfer with the interface register, which responds as a slave. Since a 16-bit register is used, it may be addressed as either a one word register or as two byte (8-bit) registers.

As shown in Figure 4-1, the basic interface uses an M105 Address Selector Module to decode the Unibus address lines and to control the clocking of information into the register and the gating of output information from the register to the bus data lines. The register is interfaced to the bus input data lines by ungated receivers, and the inputs are clocked into the register by a CLOCK DATA signal from the M105 Address Selector. The register outputs are gated through the drivers by the GATE DATA TO BUS signal. This output gating is necessary to prevent the register from affecting the Unibus data lines when the interface is not participating in a bus data transfer operation.

4.2.2 Data Transfer Operation

The read/write interface can participate in both DATI (or DATIP) and DATO (or DATOB) transfers. Whenever the processor conducts a DATO transfer to the bus address assigned to the read/write register, the data is applied through the bus receivers to the register input. At this time, both the OUT HIGH H and OUT LOW H signals are produced by the M105 Address Selector (see Figure 3-9). When MSYN is asserted by the processor, the decoded address causes the M105 to produce a SELECT 0 signal which is gated by the two OUT H signals to clock data

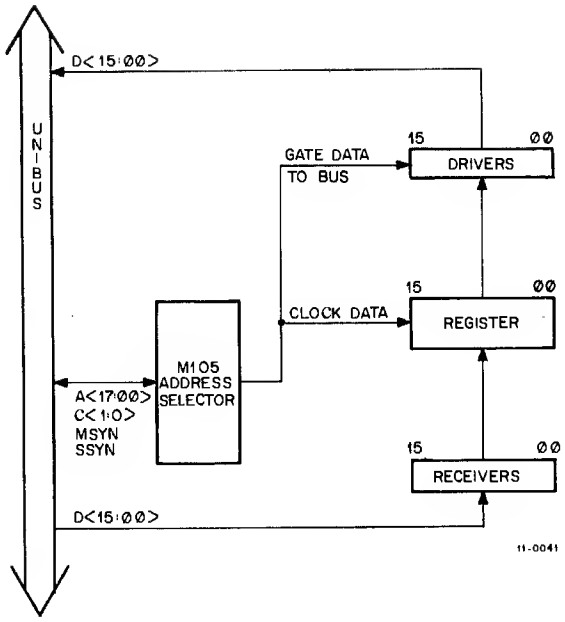


Figure 4-1. Basic Interface – Block Diagram

into the register. The Unibus timing guarantees that data is valid 75 ns prior to assertion of MSYN (refer to Paragraph 2.5). Therefore, the inputs have settled before the positive-going transition of the clock signal occurs.

A DATOB transfer functions in a similar manner, except that only one byte of the register is clocked. If address line A00 is 0, the M105 Module asserts OUT LOW H but not OUT HIGH H. If A00 is 1, then only OUT HIGH H is asserted. In either case, data is only clocked into the appropriate byte portion of the register.

When a DATI transfer occurs, the processor addresses the interface and asserts MSYN. In addition, the M105 Module asserts the same SELECT 0 signal. However, in this case, the SELECT 0 is gated by the IN rather than the OUT signals. The IN signal is generated by the state of the bus C lines. Gating of the SELECT 0 signal by the IN signal produces a GATE DATA TO BUS signal that gates the output data from the register to the Unibus. The M105 Module generates SSYN to indicate that data is ready on the output bus data lines.

4.2.3 Circuit Implementation

Figure 4-2 illustrates a possible method of implementing the circuits in the basic read/write interface. The types and quantities of modules used are: the M105 Address Selector Module (one); the M785 Bus Transceiver Module (two); the M206 General-Purpose Flip-Flop Module (three); and the M617 4-Input Power NAND Gate Module (one). The M105 and M785 Modules are described in Chapter 3 and the M206 and M617 Modules are described in the Digital 1970 Logic Handbook. The modules are interconnected and may be mounted in a BB11 System Unit as shown in the insert on Figure 4-2.

Input data flows from the Unibus, through the input gates on the M785 Modules, to the data inputs of the M206 Flip-Flops comprising the register. The gating provided on the M785 Receivers is not used, and all gates are wired to continuously receive data. Data stored in the register is protected from these changing inputs by the requirement for a clocking signal to load data into the register.

The output data from the register is gated to the bus data lines through the driver sections of the M785 Modules. The M785 Modules are used in this example because the M785 provides the exact combination of input and output gates needed for an 8-bit read/write register. When the number of receivers differs from the number of

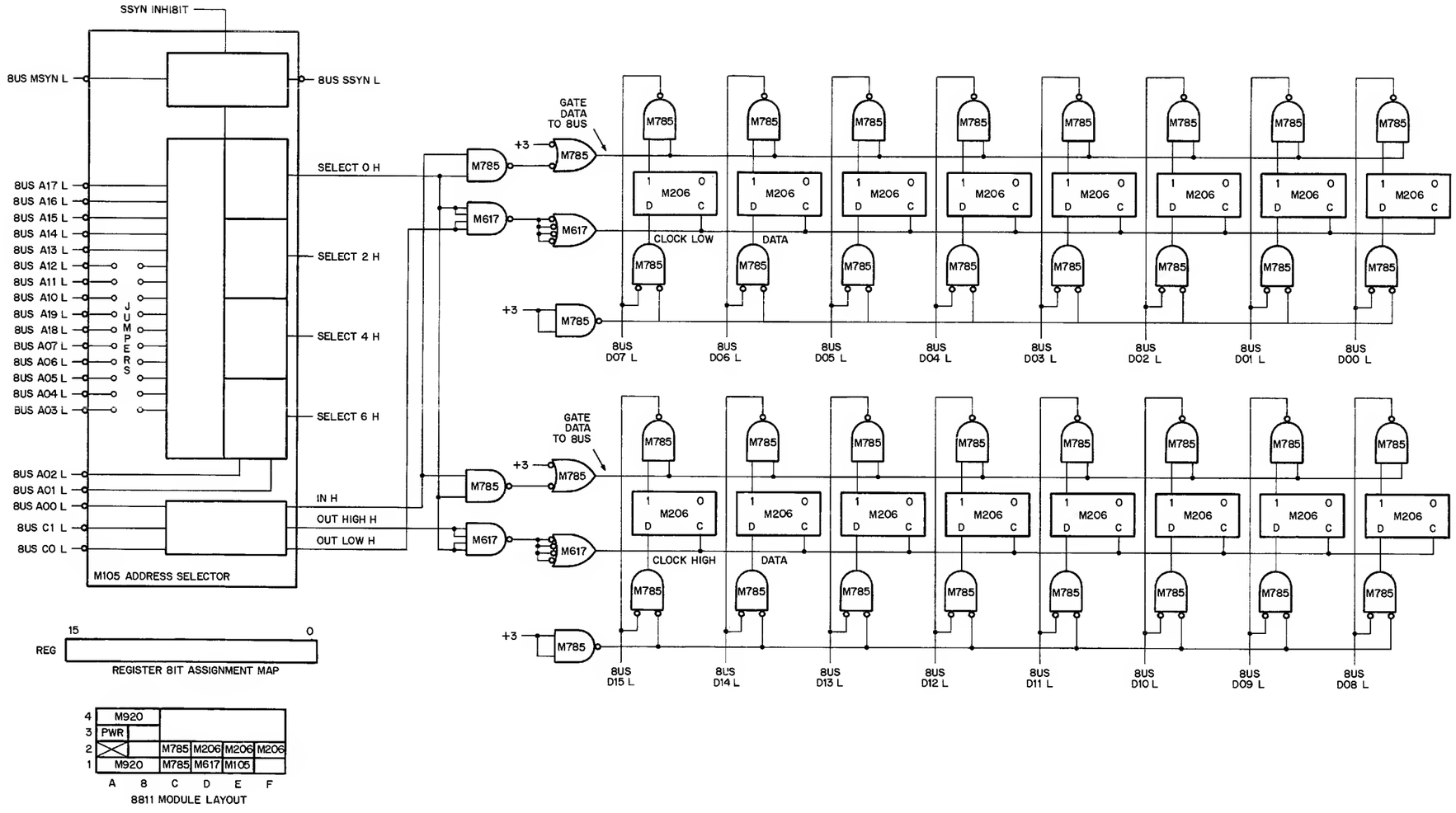


Figure 4-2. Basic Interface – Circuit Schematic

drivers required in a specific interface, combinations of M783 Bus Driver Modules and M784 Bus Receiver Modules may be used. This example is devoted to illustrating the use and interconnection of bus drivers and receivers rather than indicating the specific modules used in implementation.

4.2.4 Programming the Interface

All data transfers in the basic read/write interface are under processor control, and all memory reference instructions may directly address the interface. If the mnemonic REG is assigned to the register address, the instruction MOV REG, R4 reads the data stored in the register (a DATI operation) and places the data in general register 4 of the processor. The instruction MOV R4, REG reverses the data flow so that the data in general register 4 is placed in the interface register (a DATIP, DATO operation). Any instruction that can access a bus address can conduct data transfers with the interface register. Therefore, the contents of the register may be incremented by an INC REG instruction or summed with an arbitrary value by an ADD VALUE, REG instruction.

4.3 PROGRAMMED DEVICE INTERFACE

A circuit similar to the one in the preceding example is used as the basis for the program controlled interface to an analog-to-digital converter (ADC). It is beyond the scope of this discussion to describe the logic or internal operation of the ADC. The ADC is simply a representative example of many possible external devices that may be interfaced with a design similar to the one discussed in this section. The ADC input and output signals, however, are covered in the following paragraph because of the requirements they place on the interface.

4.3.1 Analog-to-Digital Converter

The analog-to-digital converter used in this example consists of a multiplexer and converter, (see Figure 4-3). The multiplexer selects one of 64 analog inputs and applies it to the converter, which produces the digital equivalent of the analog input.

The interface must provide seven input control signals to the ADC. One input is the start conversion signal, which is a positive transition that causes the ADC to begin the conversion process. The other six control signals are applied to multiplexer address lines so the ADMUX register can be used to select one of the 64 analog inputs.

The interface receives 11 output signals from the ADC. One of these is the conversion complete signal. When the conversion process starts, the conversion complete signal becomes 0V and remains at that level until the conversion is finished. At that time, the signal becomes +3V to indicate that the digital output reflects the analog input (the conversion is complete).

The remaining ten output lines represent the digital value equivalent to the analog input. A zero on any line is indicated by 0V and a one is indicated by +3V.

Signal levels used in the interface are standard DEC levels. Both these levels and the signal levels on the Unibus are discussed in Paragraph 3.2.2.

4.3.2 Interface Description

The program controlled interface allows the program to select a specified analog input for application to the ADC and then causes the resultant digitized output and conversion complete signal to be placed on the Unibus data lines to transfer data into the bus master.

The heavy lines in Figure 4-3 indicate logic added to the interface of the previous example. The interface functionally operates with three bus addresses. One address is assigned for the multiplexer (ADMUX) register,

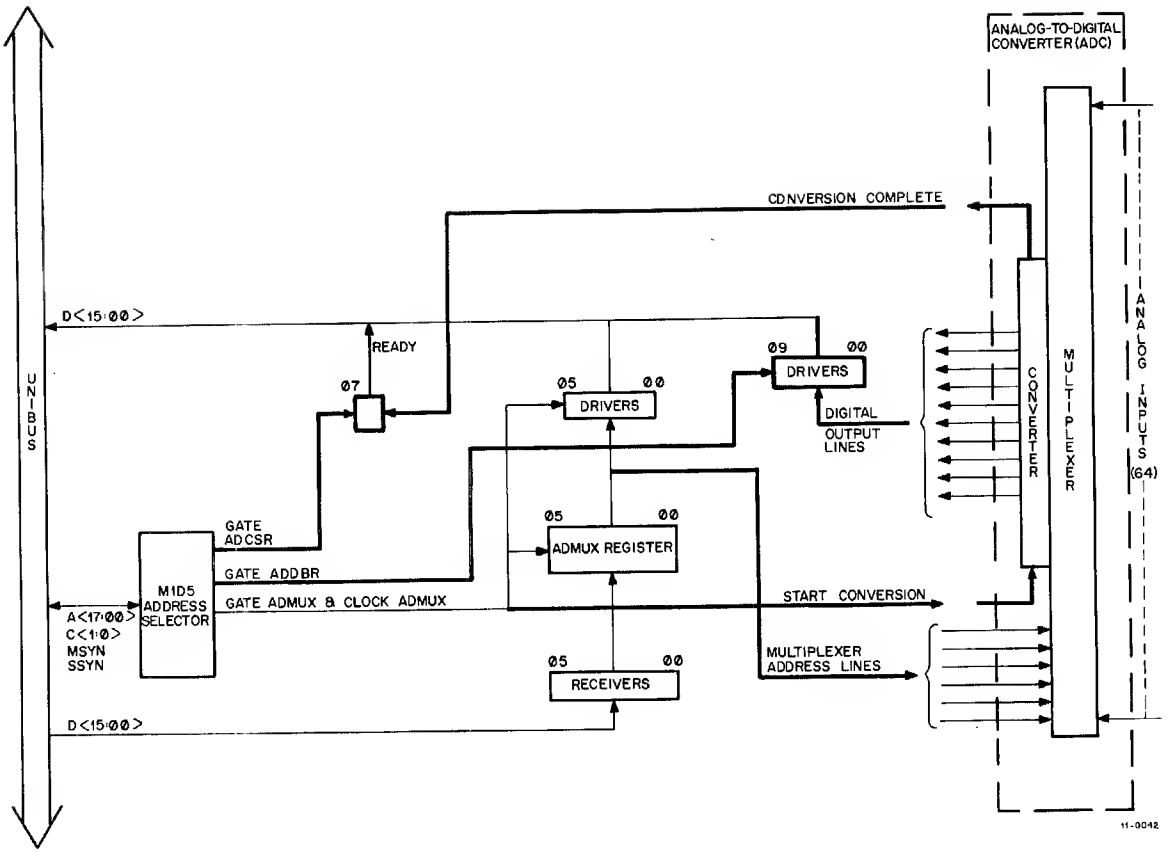


Figure 4-3. Programmed Device Interface – Block Diagram

which is similar to the register in the previous interface example. The second address is for the converted digital output (ADDBR) of a read-only register, and the third address is assigned to a 1-bit control and status register (ADCSR).

The M105 Module decodes the bus address to produce one of three select signals depending on which register is being accessed. The three select signals are gated by IN H and OUT LOW H to produce the four signals (GATE ADCSR, GATE ADDBR, GATE ADMUX and CLOCK ADMUX) shown in Figure 4-3. Only the ADMUX register accepts inputs from the Unibus through the receivers. However, the outputs of all three registers are gated to the bus through separate sets of bus drivers.

Connections between the ADC and interface may be made by a cable connector such as the M908 Module described in the Digital 1970 Logic Handbook.

4.3.3 Transfer Operations

The program controlled interface participates in bus data transfers in substantially the same manner as the basic interface described in Paragraph 4.2. Each of the three interface registers can be read during a DATI operation. In addition, the multiplexer (ADMUX) register can be loaded by a DATO operation. Although only the multiplexer register accepts data during a DATO, the other two registers respond when a DATO cycle occurs. If any of the three registers is addressed during a DATO, the M105 Module produces SSYN to complete the bus

operation. This is necessary to operate the interface with the KA11 Processor because the destination operand of all instructions that reference external data (except TST, CMP, and BIT) is transferred by a DATIP, DATO sequence of bus operations. If the interface does not respond to the DATO operation, the processor cannot continue with the program.

4.3.4 Circuit Implementation

Figure 4-4 includes a map of bit assignments for the three registers and a layout for mounting the logic modules in a BB11 System Unit. Neither the M105 Address Selector Module nor the ADC is shown on the figure, but the signals generated by these units are indicated. The connections to the Unibus can be implemented with one M785 Unibus Transceiver Module for the multiplexer register and one M783 Unibus Transmitter Module for the data and control registers. Separate gating must be supplied to use one of the four individual bus drivers on the M783 for a ready bit. The CONVERSION COMPLETE signal is renamed to READY after it passes through the bus transmitter.

4.3.5 Programming the Interface

The START CONVERSION signal, which begins the device cycle, is generated in this interface by the CLOCK ADMUX signal, which loads the multiplexer register. In normal operation, the processor loads the multiplexer register; this action starts the ADC; tests the READY (CONVERSION COMPLETE) bit until the bit is set; and then transfers the data from the digital output lines of the ADC to the processor. A possible sequence of instructions to perform this task is given below. This program selects an input, waits for the device to complete the conversion, and then transfers the result to register 4.

```
        MOV     INPUT, ADMUX    ; select analog input
READY:  TSTB    ADCSR           ; check for conversion complete
        BPL     READY           ; no, test again
        MOV     ADDBR, R4       ; yes, obtain data
```

INPUT is a location containing the number of the desired analog input line.

A subroutine to examine a series of inputs might be written as follows:

```
MUXSCN: MOV     BUFADR, R4      ; initialize data pointer
        CLR     ADMUX           ; select input line zero
LOOP:   TSTB    ADCSR           ; check for conversion complete
        BPL     LOOP            ; no, test again
        MOV     ADDBR, (4)+     ; yes, place data in buffer
        CMP     ADMUX, #77      ; last line?
        BEQ     DONE            ; yes, go to done
        INC     ADMUX           ; no, go to next input
        BR      LOOP            ; go to loop
DONE:   RTS     R7              ; exit from subroutine
```

where: BUFADR is a location in core containing the address of the first word of a 64-word buffer
ADCSR is the interface status register
ADMUX is the multiplexer register
ADDBR is the data register

This subroutine is called by the instruction: JSR 7 MUXSCN. The subroutine initializes general register 4 as a pointer to the buffer; initializes the multiplexer register to zero; and sequentially reads the 64 inputs into the corresponding buffer location. When each input has been read once, control returns to the calling program with the contents of general register 4 as the address of the word after the last word of the buffer.

Since loading the multiplexer register starts operation of the device cycle, ADMUX should not be accessed as a destination operand except by a TST, BIT, or CMP instruction. In addition, the INC ADMUX instruction should follow the CMP instruction. This avoids initiating unwanted device operation and allows the subroutine to be immediately re-called.

4.4 INTERRUPT SERVICED INTERFACE

The interface to an analog-to-digital converter would be more versatile if it included an interrupt capability. An interrupt serviced interface with this capability can be formed simply by adding an M782 Interrupt Control Module and one bit to one of the registers in the programmed device interface described in Paragraph 4.3.

The interrupt serviced interface allows the processor to concurrently execute instructions of another program while the analog-to-digital converter (ADC) performs a cycle of operation. The processor responds to a READY (CONVERSION COMPLETE) signal from the ADC by interacting with the device and analyzing the data after it has been collected. This interface eliminates requiring the processor to spend time testing for a ready signal, such as in the case of the programmed device interface.

Whenever a device interface is required, the designer must compare the cost of additional interrupt hardware with the device requirements in terms of transfer speed, frequency of transfers, and amount of use, to determine whether a programmed device interface or interrupt serviced interface is more economical.

4.4.1 Interface Description

Figure 4-5 is a block diagram of the interrupt serviced interface which consists of the programmed device interface with the addition of an M782 Interrupt Control Module, one flip-flop, and one bus driver. This interface can operate either in the same manner as the interface described in Paragraph 4.3 or in an interrupt mode. The additional flip-flop is used to enable or disable interrupt operations. If the flip-flop (which is bit 6 of the control status register) is set by the program, the CONVERSION COMPLETE signal from the ADC causes the M782 Interrupt Control Module to initiate an interrupt. The interrupt operation is described in Paragraph 2.4.2.

4.4.2 Circuit Implementation

Figure 4-6, a circuit schematic of the interrupt serviced interface, also includes a bit assignment map for the three registers and a module layout of a BB11 System Unit. The M782 Module is represented by two blocks similar to the interconnection diagram shown in Figure 3-13. The ADC and the M105 Module are represented by the signal lines connecting them to the logic.

One possible method of implementing the circuit for this interface is to add an M206 General-Purpose Flip-Flop Module and an M782 Interrupt Control Module to the implementation of the programmed device interface described in Paragraph 4.3.4. The M206 Module provides the necessary flip-flop for bit 6 of the control status register, which is used as the interrupt enable bit. The function of the M782 is discussed in the following paragraph. Implementation is illustrated by the module layout in Figure 4-6.

When the ADMUX register is loaded by a DATO or DATOB bus operation, the CLOCK ADMUX signal also starts a cycle of ADC operation. If the interrupt enable flip-flop is set (by loading a one into bit 6 of the control status register by means of a DATO or DATOB operation), then the M782 initiates a bus request when the CONVERSION COMPLETE H (READY) signal is asserted. Although Figure 4-6 indicates that this is a level 5 priority request, any desired bus request (BR) level may be used. When the request is acknowledged, the interface conducts an INTR bus operation. The vector address transmitted to the processor is selected by jumpers on the M782 Module. The program normally responds to the interrupt by collecting the digital output from the ADC and then initiating another conversion cycle. The last conversion cycle is serviced by collecting the data and then transferring to an evaluation routine.

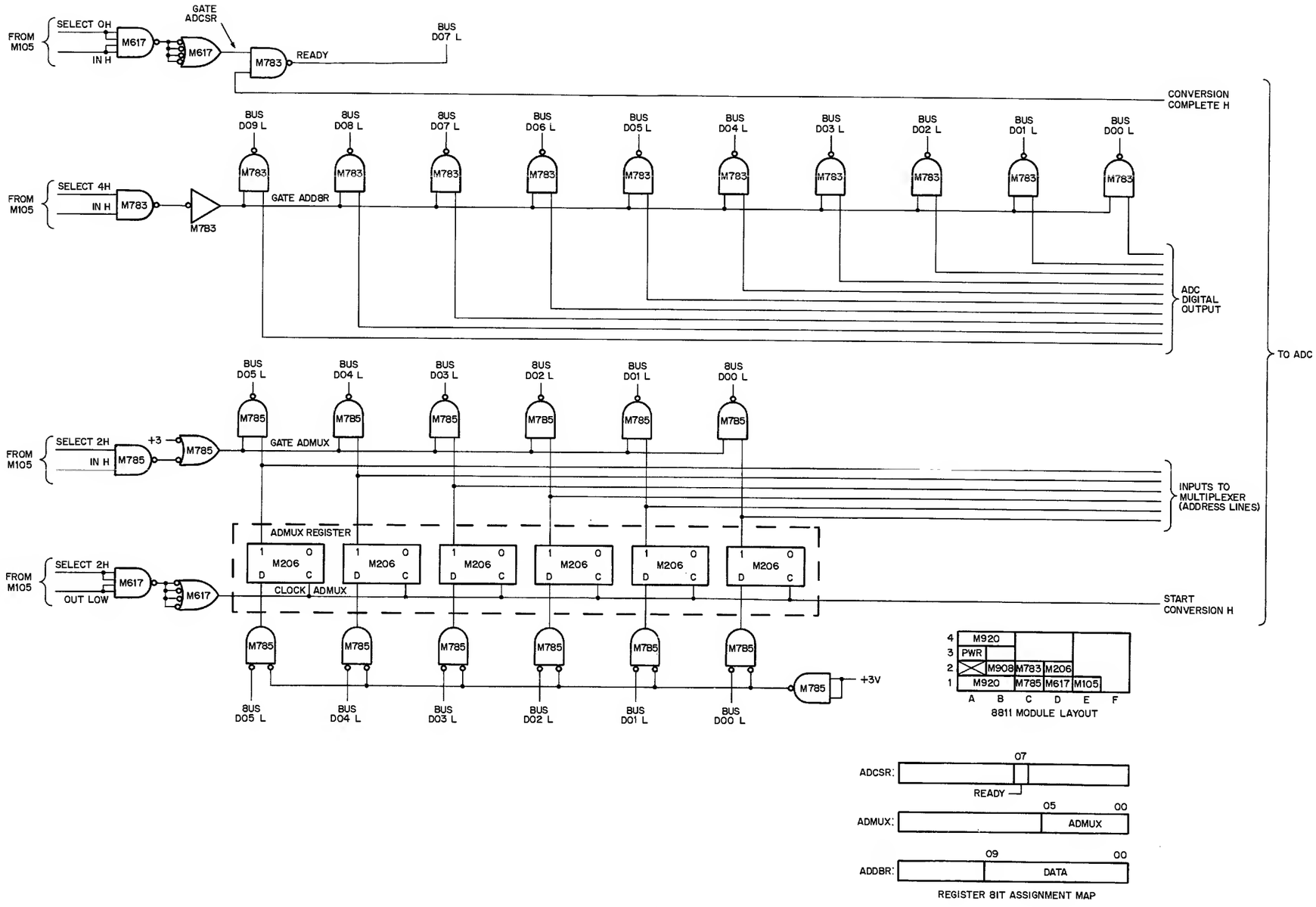


Figure 4-4. Programmed Device Interface -- Circuit Schematic

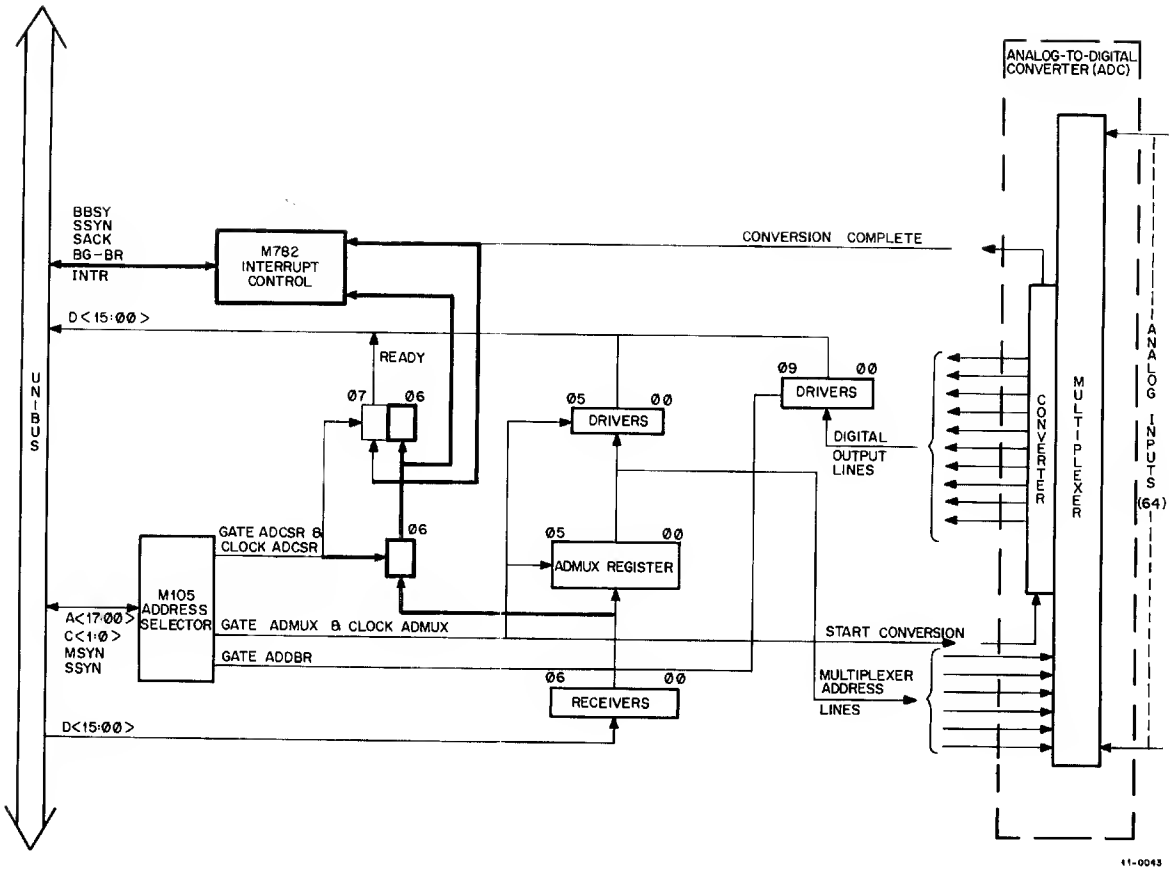


Figure 4-5. Interrupt Serviced Interface – Block Diagram

4.4.3 Interface Programming

The following program is a typical interrupt service routine that collects data from the ADC and enters an evaluation routine after the final conversion cycle.

```
ADCVEC:   ADCSER                    ; set up ADC vector area
          240                       ; status includes priority level 5
            .
            .
            .                       ; main program follows
          MOV BUFSTRT, BUFADR       ; initialize buffer pointer
          CLR ADMUX                 ; start multiplexer at channel 0
          MOV #100, ADCSR           ; enable interrupt
            .
            .
ADCSER:   MOV ADDBR, @BUFADR        ; collect data
          CMP BUFADR, BUFSTRT+174   ; last one?
          BEQ DONE                  ; yes, go to DONE
          ADD #2, BUFADR            ; no, increment pointer
          INC ADMUX                 ; increment multiplexer and
                                    ; start conversion
          RTI
```

```
DONE:   CLR ADCSR                   ; clear interrupt enable
        .
        .                           ; follow this with the
                                    ; evaluation routines
```

where: ADCSR, ADMUX, and ADDBR are the device registers in the interface
BUFSTRT contains the starting address of a buffer
BUFADR is a location to be used by the device service routine
ADCVEC is the address specified by jumpers on the M782 Module and contains the address of the device service routine tagged ADCSER
ADCSER device service routine

After the initiation instructions in the main program are executed, the interrupts cause the processor to execute the ADCSER routine. The last time this is performed, the evaluation routine is also executed.

The CLR ADMUX instruction should precede the MOV #100, ADCSR instruction to prevent the interface from causing an immediate interrupt, which could occur if the interrupt enable bit is set when the device has the conversion complete (ready) signal asserted.

For longer service routines, in which the usc of registers is desirable to increase access speed to certain words, the previous register contents could be pushed onto the processor stack at the beginning of the routine and popped back in the general register before returning to the main program. Registers should not be used in interrupt service routines without this procedure because the use of the general registers by the main program is not predictable. The service routine is not much chorter than the programmed device routine in the previous example; however, it does have the major advantage of not tying up the processor while the device is performing an operation cycle.

If the evaluation routine is to return control to the main program at the same point in the service routine, this may be accomplished by terminating the evaluation routine with an RTI instruction. If any other type of return is used, the program must remove the old PC and PS that were placed on the stack by the interrupt operation. Removal is accomplished by executing an ADD #4, R6 instruction.

4.4.4 DR11-A Implementation

A more convenient method of implementing an interrupt serviced interface is to use an M786 General Interface Module to make a DR11-A 16-Bit General Interface. Figure 4-7, a layout of the modules mounted in a DD11-A System Unit, shows the savings in space and interconnections compared to the implementation method discussed in Paragraph 4.4.2. The DD11-A System Unit is prewired to accept four DEC small peripheral interfaces; e.g., DR11-A. A discussion of the DR11-A, including specifications, is presented in Paragraph 3.3.6.

Figure 4-7 is similar to Figure 4-5 because the DR11-A logic is used in the same manner and with the same programs as any other logic used to implement an interrupt serviced interface. The M786 Module portion of the DR11-A provides cable connectors; therefore, no additional wiring or connectors are required.

Connections between the ADC and the M786 Module are made as follows:

CONNECTOR	DR11-A	ADC
1	OUT (06:00)	Multiplexer inputs
	NEW DATA READY	Start conversion
2	IN (09:00)	Digital outputs
	REQUEST B	Conversion complete

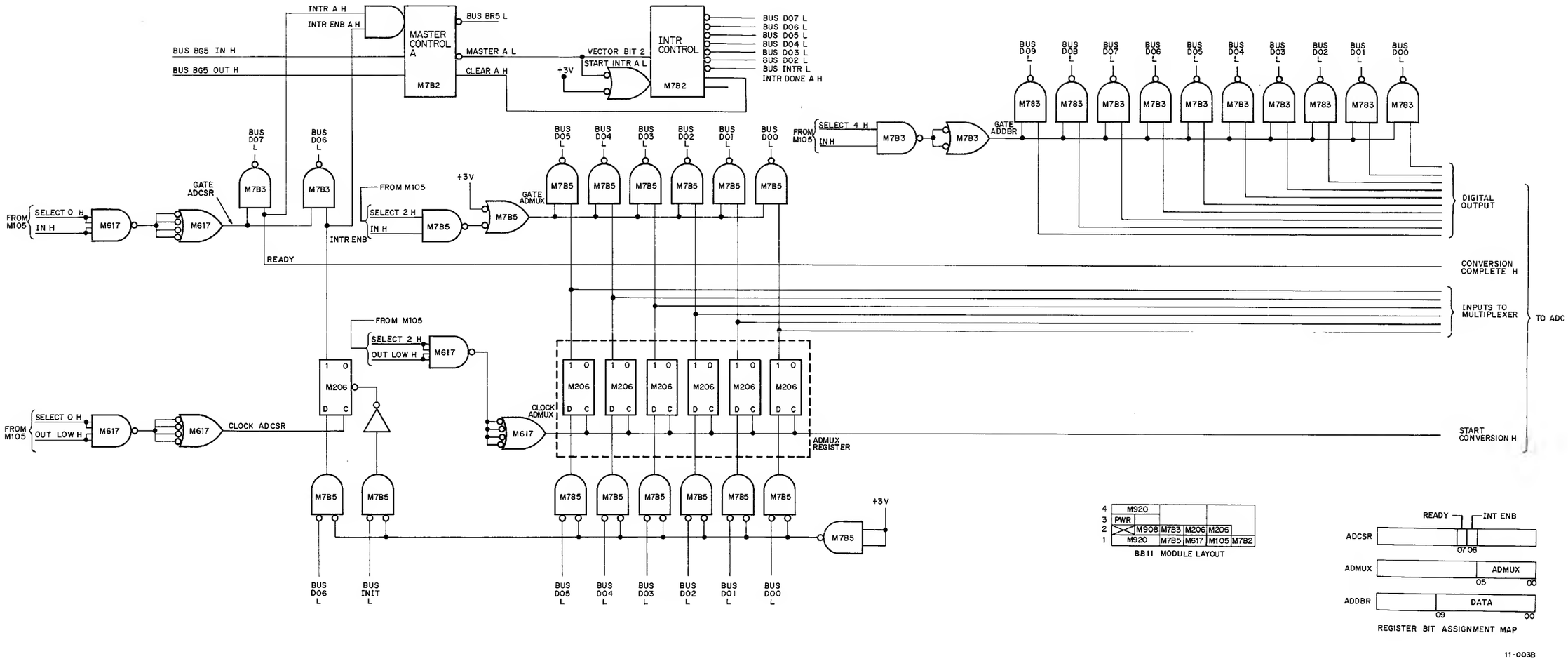


Figure 4-6. Interrupt Serviced Interface – Circuit Schematic

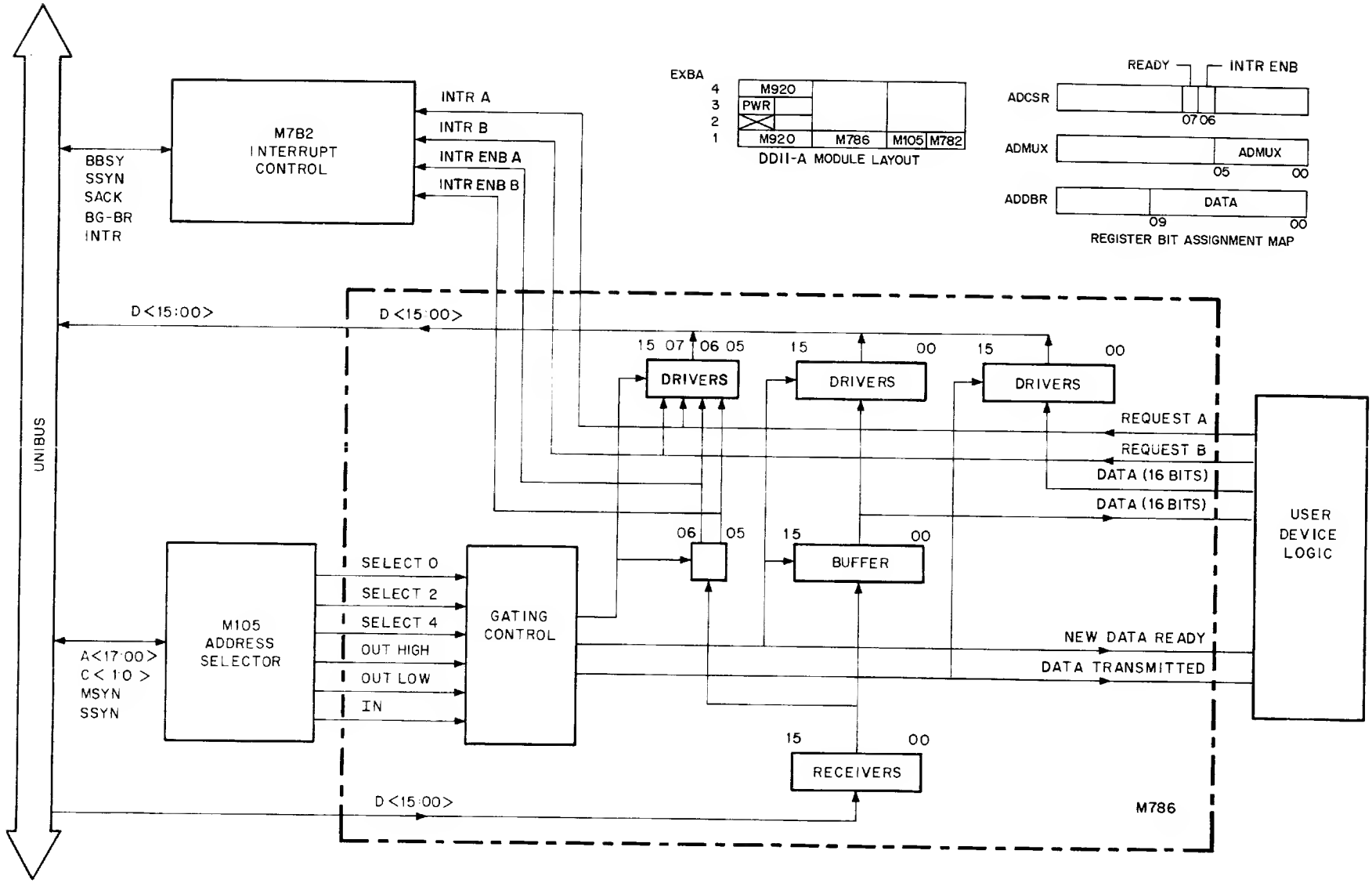


Figure 4-7. DR11-A Implementation – Block Diagram

4.5 DIRECT MEMORY ACCESS (DMA) INTERFACE

The direct memory access (DMA) interface conducts data transfer operations to place data from the device directly into memory. A DMA interface performs a large number of transfers without any processor intervention, thereby reducing program and execution time overhead. When the interface device registers are initialized, all transfers take place under control of the interface, thereby eliminating processor time. The processor is notified by an interrupt when all the data has been transferred and the program responds appropriately.

Figure 4-8 is a block diagram of a DMA interface for the ADC. The DMA is designed by adding circuits to the interrupt serviced device interface.

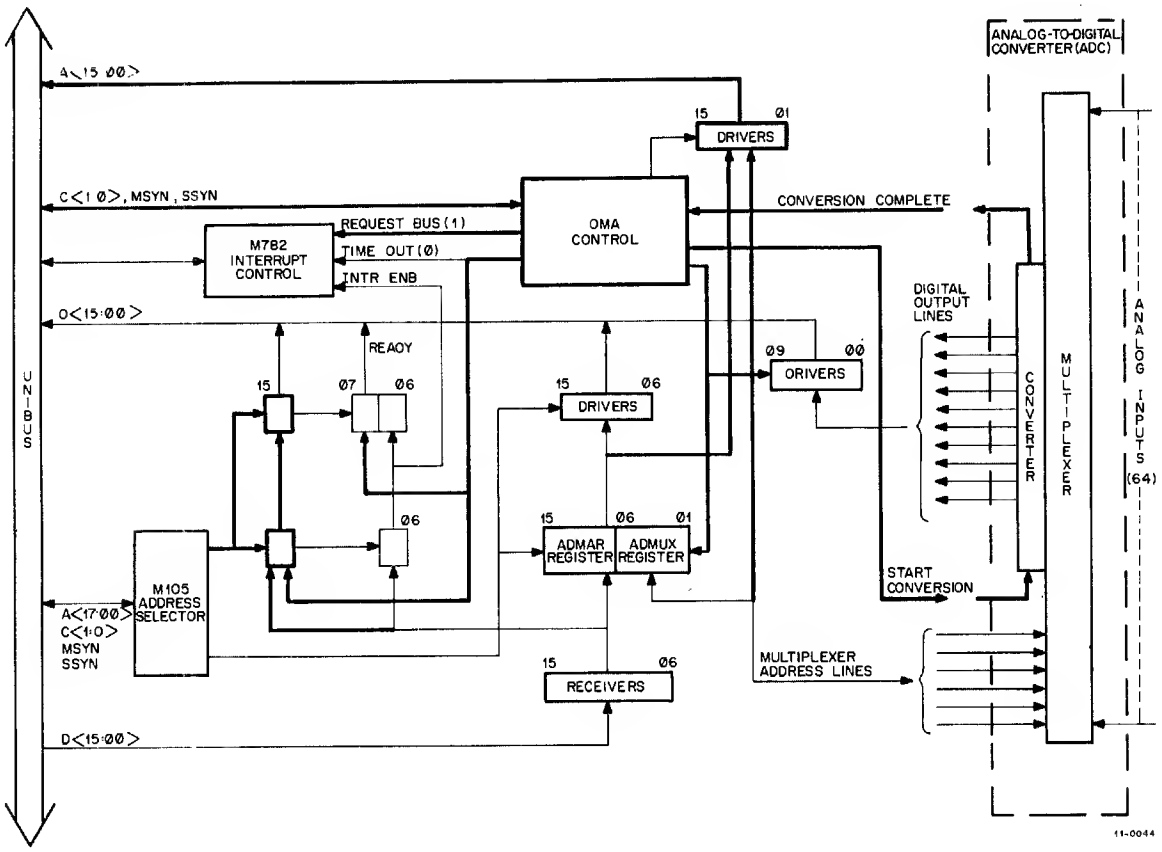


Figure 4-8. DMA Interface – Block Diagram

4.5.1 Interface Description

Interface operation begins when the program loads the memory address register (ADMAR) with the address of the first memory location where data is to be stored. The interface starts an ADC conversion cycle. When the digital data is available from the ADC, the interface requests bus use by asserting an NPR request. When the device becomes bus master, it transfers the data to core memory. Completion of the bus transfer causes the multiplexer register (ADMUX) to be incremented, thereby selecting the next input channel. The multiplexer register is part of the memory address register; therefore, the next core memory location is also selected. At this

point, a new conversion cycle begins. This process is repeated until each input channel is read and the digital data is stored in a core memory location. The interface then sets the ready flip-flop, which causes an interrupt.

4.5.2 Interface Implementation

The DMA interface is constructed by adding one set of bus drivers and the bus transfer control logic to the interrupt serviced interface; therefore, the functions assigned to the registers differ in this case, and implementation differs accordingly. The multiplexer register, expanded to 15 bits, also serves as a memory address register. Nine of these bits (15:07) are under program control and serve as a base address for a series of locations used as a data collection buffer by the interface. The remaining six bits (06:01) are implemented as a counter that steps through the 64 inputs and also addresses 64 successive word locations in the core memory. The six multiplexer bits are not accessible from the bus and cannot be read nor altered by the program; therefore, transfers always start on 64 word boundaries.

The interface uses an interrupt to signal completion of the series of transfers. The interrupt enable (INTR ENB) and READY bits of the CSR operate similar to the interrupt serviced interface.

Figure 4-9 is a circuit schematic of the DMA interface, which is implemented entirely with standard Digital M-series modules described in the Digital 1970 Logic Handbook. Time delays are provided by M302 Dual Delay Multivibrator Modules used as one-shots. The multiplexer (ADMUX) counter is implemented by an M211 Binary Up/Down Counter Module.

Loading the ADMAR register (CLOCK ADMAR) also clears the multiplexer counter and the READY flip-flop, thereby initiating a conversion cycle by causing START CONVERSION H to become asserted.

When the conversion is complete, the CONVERSION COMPLETE H signal sets the REQUEST BUS flip-flop, which causes the M782 Interrupt Control Module to assert an NPR request. When bus control is granted, the M782 asserts BBSY on the Unibus and asserts the MASTER A L signal. This signal asserts line C1 on the Unibus to force a DATO operation; gates the information from the ADMAR and ADMUX registers to the A⟨15:01⟩ lines; and gates the digital data to the D⟨07:00⟩ lines. The MASTER A L signal also starts a one-shot causing MSYN to be asserted 150 ns after the A, C, and D lines.

When the slave device responds by asserting SSYN, MSYN is cleared. This starts a timing chain consisting of two one-shots. After 75 ns, the first one-shot triggers the second one-shot. The leading edge of the pulse (END TRANSFER H) clears the REQUEST BUS flip-flop. This action causes the M782 to clear both the BBSY and MASTER A L signals, thereby clearing the A, C, and D lines. The END TRANSFER H signal also serves as the count input (COUNT IN) to the multiplexer counter (ADMUX). After 600 ns, the second one-shot times out and its output (END CYCLE L) returns to a low (0V) level. If the READY flip-flop has not been set by a count overflow from the ADMUX counter, START CONVERSION H is asserted to start the next conversion cycle. If, however, the ADMUX counter has overflowed and set the READY flip-flop, no ADC operation is started and an interrupt bus request is made. A 10-μs time-out circuit is provided to protect against failure of the slave device to acknowledge the transfer with a SSYN signal (refer to Paragraph 2.5.2). The 10-μs one-shot is triggered when MSYN is asserted. If the trailing edge of the CLOCK TIME OUT L signal occurs while the MSYN flip-flop is set, the TIME OUT flip-flop is set. The resultant TIME OUT (1) H signal disables the request bus input to the M782, thereby releasing the bus; sets the READY flip-flop to inhibit further conversion cycles; and initiates an interrupt with the M782 Master Control B Channel. The status of the time-out (which is an error condition) is available in bit 15 of the status register. Clearing bit 15 clears the time-out error.

The modules required to implement this interface fit into one BB11 System Unit. All interface modules, including the M782 Interrupt Control, M105 Address Selector, and a device cable connector, can be inserted into

the logic slots of one system unit containing power and Unibus connectors. Thc BB11 System Unit is described in Paragraph 3.4.1. Materials for the entire interface cost less than $750.

4.5.3 Interface Data Flow

Interface operation begins when data is loaded from the bus into the ADMAR register. This operation clears the ADMUX counter and initiates operation of the ADC. When the ADC completes an operation, the interface control transfers data from the digital output lines of the ADC to the bus data lines. At the same time, the interface control gates the ADMAR register (including the ADMUX counter) outputs to the bus address lines. When the ADMUX counter overflows, the interface control initiates an interrupt operation.

Thc following is an instruction sequence to initiate device operation:

```
MOV  BUFADR, ADMAR        ; load address and start
MOV  #100, ADCSR          ; enable interrupt
```

where: BUFADR is the address of the first word of a buffer and must have all zeros in bits 0 through 6

The interrupt routine for this interface is equivalent to the data evaluation routine suggested in the interrupt serviced interface. Tlie routine should begin with a CLR ADCSR instruction to disable further interrupts and should terminate with an RTI instruction.

The ADMAR register can be read as a source operand without spurious clocking of the device operation cycle, but the ADMUX counter is not accessible from the bus.

Thc interrupt enable flip-flop (bit 6 of the ADCSR) is entirely under program control but the time-out flip-flop is set by time-out error conditions in the interface. The ready bit of the ADCSR (bit 7) is not under program control. It may be read by the program but cannot bc altered except by initiating operation of the device.

4.5.4 Interface Operation Timing

Figure 4-10 illustrates the timing relationships among signals in the DMA interface. The curved lines indicate the changes in signal level that generate the indicated results.

4.6 OUTPUT INTERFACE WITH INTERRUPT CONTROL

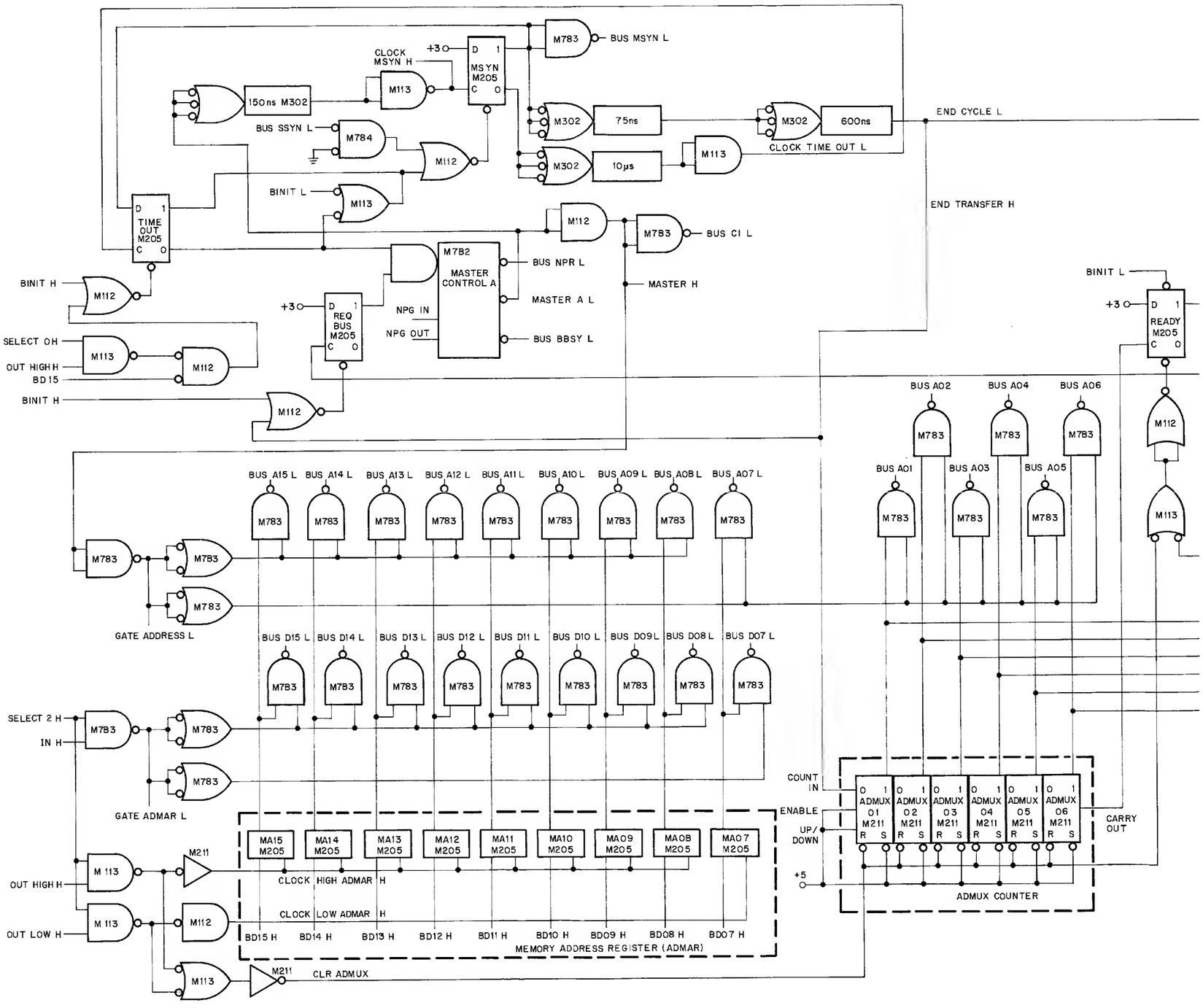
Preceding examples have illustrated various types of interfaces for a peripheral device, which provides inputs to the Unibus data lines. This example, as well as the example in Paragraph 4.7, covers interface design for a device that accepts Unibus outputs. The device shown is meant to be typical of output devices which may be interfaced by designs similar to the following examples.

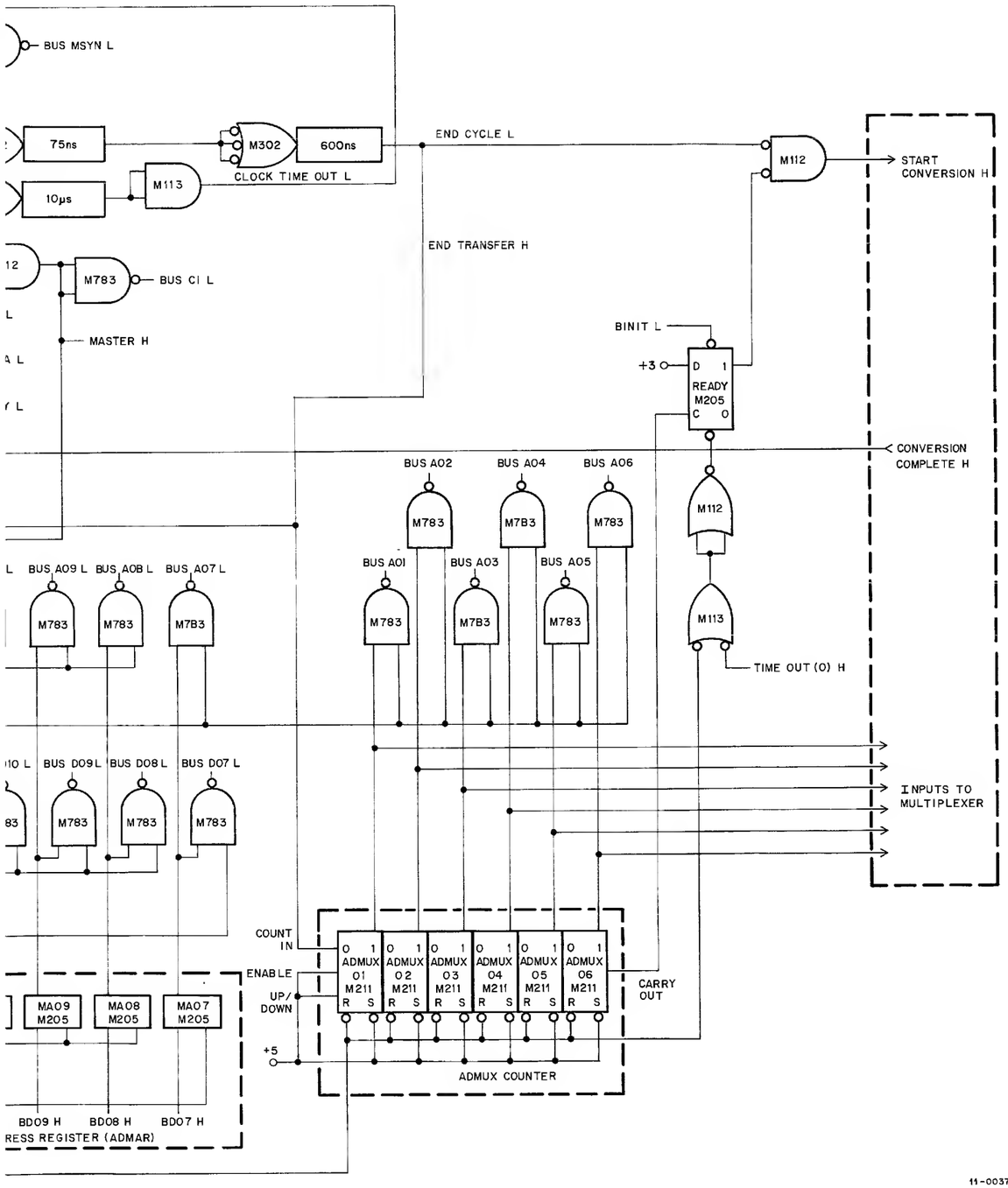
4.6.1 Device Description

A digital-to-analog converter (DAC) is a device that accepts Unibus outputs. The DAC converts a digital value to an equivalent analog voltage. The device is single buffered and the analog output follows the digital output.

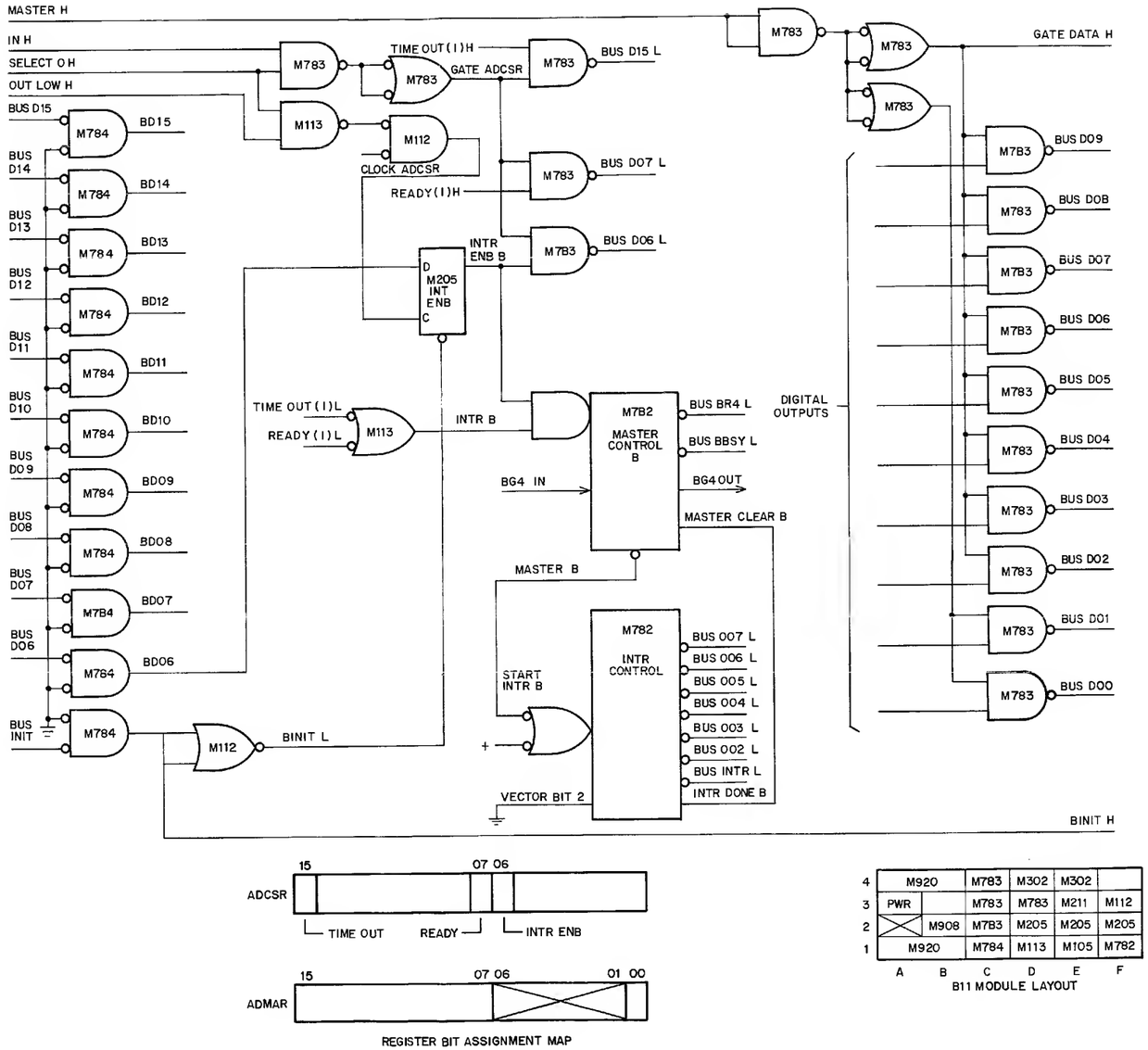
The interface provides 10 binary level inputs to the DAC. These inputs represent the digital value equivalent to the analog voltagc desired as an output. The binary levels are 0V for binary 0 and +3V for binary 1.

The DAC provides an update request output signal for the interface. This signal requests a new digital input from the interface. At intervals determined by the DAC, a high level (+3V) pulse is provided as the update request signal. This level remains low (0V) between pulses.


M783
BUS MSYN L
CLOCK MSYN H
+3
D 1
MSYN M205
C O
M113
150ns M302
BUS SSYN L
M784
M112
M302
75ns
M302
600ns
END CYCLE L
M302
10µs
M113
CLOCK TIME OUT L
BINIT L
M113
END TRANSFER H
D 1
TIME OUT M205
C O
M112
M783
BUS CI L
M7B2
MASTER CONTROL A
BUS NPR L
MASTER A L
MASTER H
BINIT H
M112
+3
D 1
REQ BUS M205
C O
NPG IN
NPG OUT
BUS BBSY L
SELECT OH
OUT HIGH H
BD15
M113
M112
BINIT H
M112
BINIT L
+3
D 1
READY M205
C O
M112
M113
BUS A02
BUS A04
BUS A06
BUS A01
BUS A03
BUS A05
BUS A15 L
BUS A14 L
BUS A13 L
BUS A12 L
BUS A11 L
BUS A10 L
BUS A09 L
BUS A08 L
BUS A07 L
M783
GATE ADDRESS L
BUS D15 L
BUS D14 L
BUS D13 L
BUS D12 L
BUS D11 L
BUS D10 L
BUS D09 L
BUS D08 L
BUS D07 L
SELECT 2 H
IN H
GATE ADMAR L
COUNT IN
ENABLE
UP/ DOWN
+5
0 1
ADMUX 01 M211
R S
ADMUX 02 M211
ADMUX 03 M211
ADMUX 04 M211
ADMUX 05 M211
ADMUX 06 M211
CARRY OUT
ADMUX COUNTER
MA15 M205
MA14 M205
MA13 M205
MA12 M205
MA11 M205
MA10 M205
MA09 M205
MA08 M205
MA07 M205
M211
M 113
OUT HIGH H
CLOCK HIGH ADMAR H
CLOCK LOW ADMAR H
M 113
M112
OUT LOW H
BD15 H
BD14 H
BD13 H
BD12 H
BD11 H
BD10 H
BD09 H
BD08 H
BD07 H
MEMORY ADDRESS REGISTER (ADMAR)
M211
M113
CLR ADMUX



11-0037



REGISTER BIT ASSIGNMENT MAP

4	M920		M783	M302	M302	
3	PWR		M783	M783	M211	M112
2		M908	M7B3	M205	M205	M205
1	M920		M784	M113	M105	M782
	A	B	C	D	E	F

B11 MODULE LAYOUT

Figure 4-9. DMA Interface – Circuit Schematic

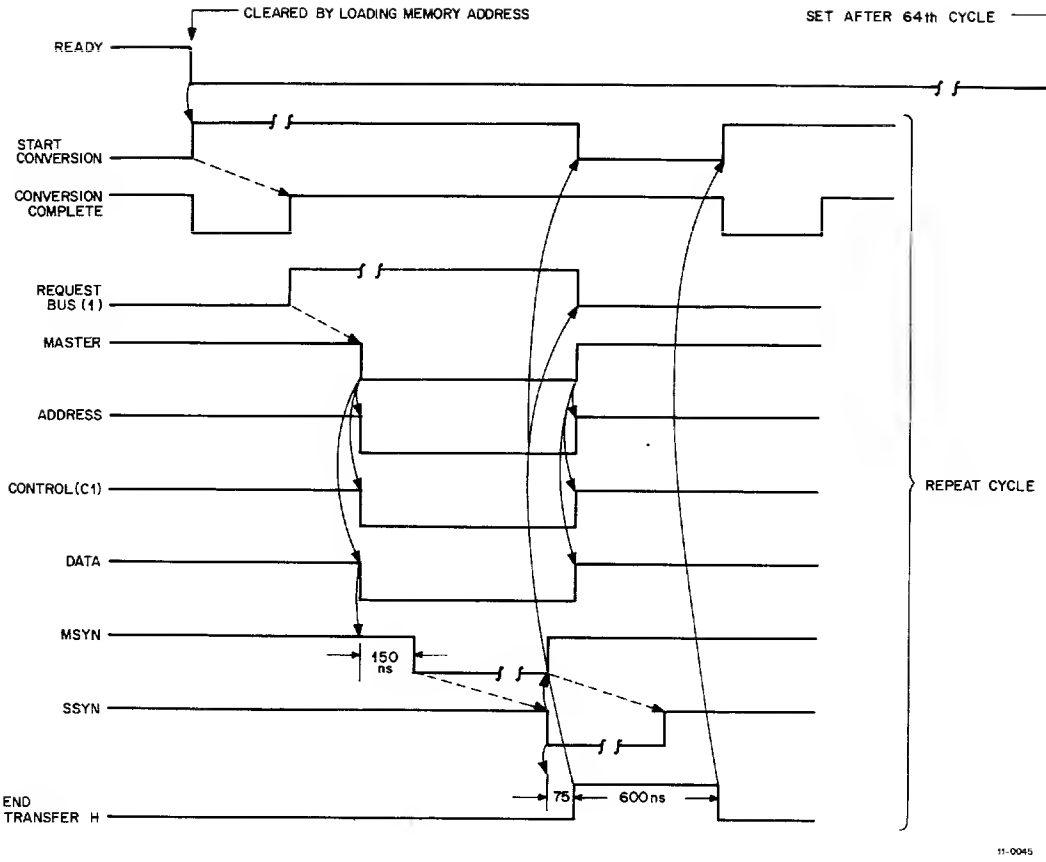


Figure 4-10. DMA Interface – Timing Diagram

4.6.2 Interface Description

The output interface with interrupt control provides a buffer register for outputs to the DAC and an interrupt control to service the DAC with an interrupt service routine. Figure 4-11 is a block diagram of the output interface.

The interface consists of two registers, an M105 Address Selector Module, an M782 Interrupt Control Module, bus receivers, and two sets of bus drivers. The two registers are the data buffer register (DADBR) and the control status register (DACSR). The request bit (bit 7) of the DACSR can be read by the bus but cannot be loaded directly from the bus. All other register bits are under direct bus control.

4.6.3 Interface Operation

When the Unibus addresses the data buffer register during a DATO transfer, the interface clocks the information from the bus data lines into the register, which then applies the information to the DAC as the 10 binary level inputs. At the same time data is clocked into the register, the REQUEST flip-flop (bit 7 of the DACSR) is cleared. After this transfer is complete, when the peripheral device requests another value, the REQUEST flip-flop is clocked high by an UPDATE REQUEST signal from the DAC. If the interrupt enable flip-flop (bit 6 of the DACSR) is set, the interface asserts the bus request line. On becoming bus master, the interface performs an interrupt operation to transfer program control to a service routine. This routine loads new data into the buffer register and then returns control to the interrupted program.

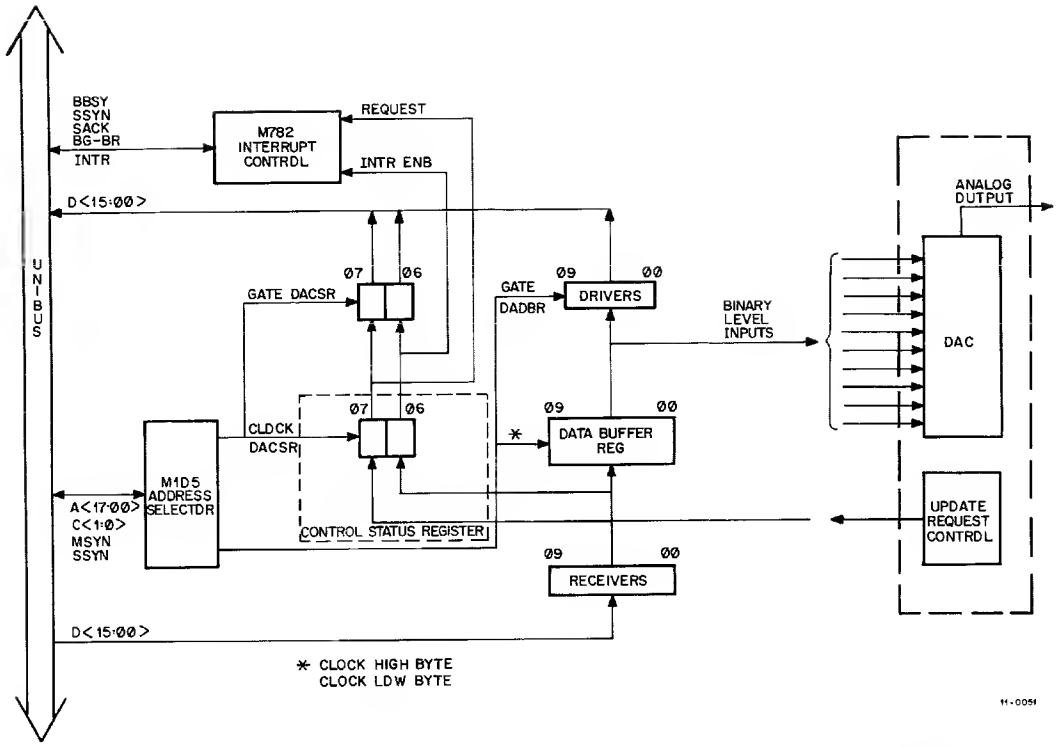


Figure 4-11. Output Interface with Interrupt Control – Block Diagram

4.6.4 Circuit Implementation

Figure 4-12 shows the output interface with interrupt control, a bit assignment map for the registers, and a module layout of a BB11 System Unit. The interrupt control is similar to the control described in Paragraph 4.4. The buffer register is identical to the register described in Paragraph 4.2.

The only additional control logic required is the request flip-flop, which initiates a cycle of interface service and remains set until cleared by the loading of DADBR.

During normal operation, data is loaded into the buffer register and transferred to the peripheral device. When an UPDATE REQUEST from the DAC starts an interface cycle, the interrupt vector is transferred to the processor. The processor again initiates the data flow by transferring a new word of data into DADBR.

4.6.5 Interface Programming

The programs described in this paragraph cause the DAC to output a time-varying signal by loading the DADBR with an initial value and then changing that value by small increments until it reaches a final value determined by the program. The analog output is 100 cycles of a triangular waveform (actually, a stepped triangular waveform) with the slope of the ascending portion equal to half the slope of the descending portion. The period of the waveform is 150 times the period between update request pulses.

4.6.5.1 **Significance of Programs** – In the interface program, the DAC output is reset to a higher value by the ADD #10, DADBR instruction or reset to a lower value by the SUB #20, DADBR instruction. In either case, the value in the DADBR is read, modified by an arithmetic operation, and the new value is stored in the DADBR. All these operations are under processor control.

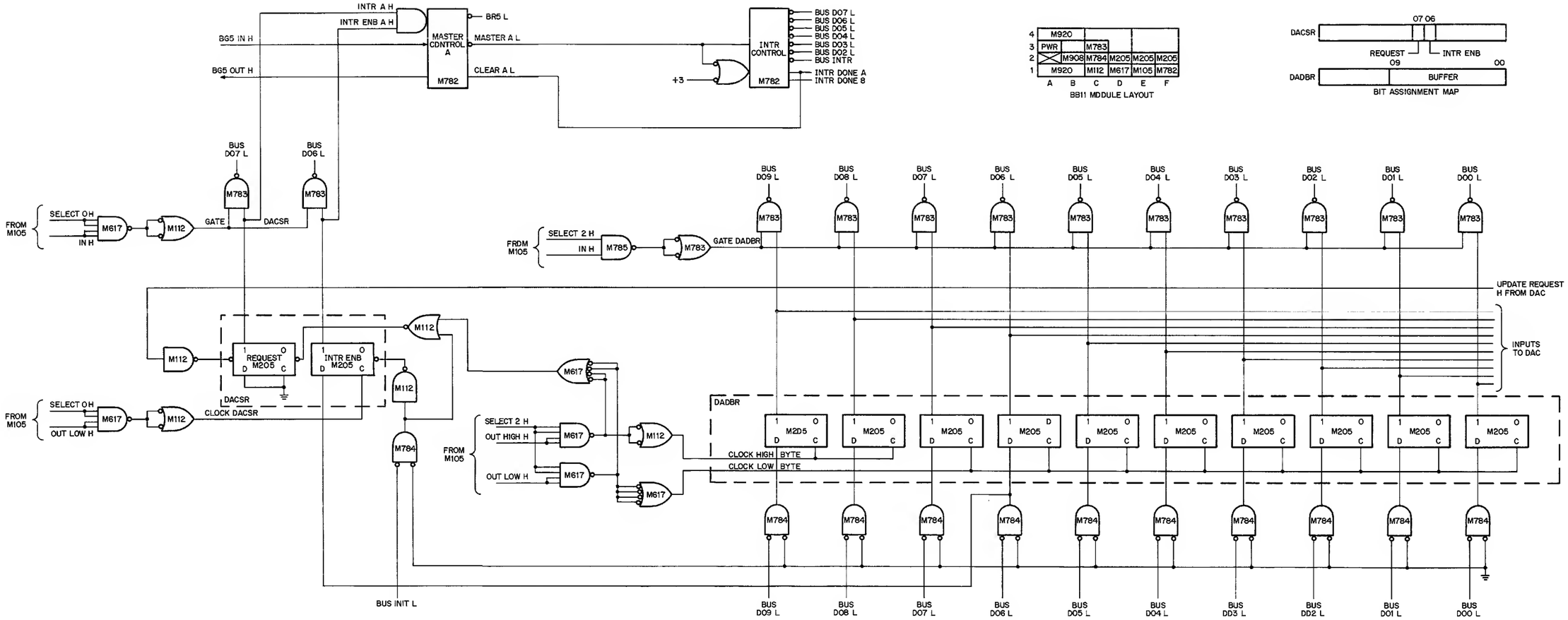


Figure 4-12. Output Interface with Interrupt Control – Circuit Schematic

The ability of the Unibus to access device registers as though they were core memory locations allows the processor to directly perform tests and modifications on the device register. This program compares the value in the DADBR with the test values. The program uses a minimum of stored data because it is not necessary to use memory locations for counters or storage of temporary values.

4.6.5.2 **Program Description** – The processor initializes operation by executing the following sequence of instructions:

```
        CLR     DADBR           ; clear data buffer register
        CLR     DASW            ; reset up/down switch
        MOV     #144, DACNT     ; set cycle counter to 100
        MOV     #100, DACSR     ; set interrupt enable
```

The interrupt service routine includes the following instructions:

```
DAVEC:  DASERV                  ; pointer to service routine
        240                     ; processor priority = 5

DASERV: TST     DASW            ; switch set?
        BPL     UP              ; no, go up
        SUB     #20, DADBR      ; yes, go down
        BNE     CONT            ; output value equals zero?
        CLR     DASW            ; yes, reset switch
        DEC     DACNT           ; reduce count by one
        BNE     CONT            ; count equals zero?
        CLR     DACSR           ; yes, disable interrupt and exit
        RTI

UP:     ADD     #10, DADBR      ; output value goes up
        CMP     DADBR, #1000    ; 1000 is top limit on value
        BNE     CONT            ; does value equal top limit?
        COM     DASW            ; yes, set switch

CONT:   RTI                     ; exit for intermediate values
```

4.7 DAC-DMA INTERFACE

A direct memory access (DMA) interface designed for a digital-to-analog converter (DAC) conducts data transfer operations from the core memory to the DAC. A DMA interface can perform a large number of transfers without any processor intervention.

The interface in this example has been designed by adding logic circuits to the interface described in Paragraph 4.6. The interface output is a sequence of digital values, which are converted by the DAC to produce a sequence of 64 analog values. The output rate is controlled by the DAC, which issues a request each time another output is desired. Prior to operation, a 64-word buffer is loaded into core memory. These words represent the digital equivalent of the desired analog outputs. During operation, a memory address register (MAR) is loaded with the bus location of the first word in the buffer. At the desired time, this word passes through the interface into the DAC which converts it to the desired analog voltage.

4.7.1 Interface Description

Figure 4-13 shows the DAC-DMA interface. The following logic has been added to the output interface with interrupt control: a DMA control, a data buffer register (which is not accessible from the Unibus), and a set of bus address line drivers on the outputs of the program controlled register. The program controlled register serves

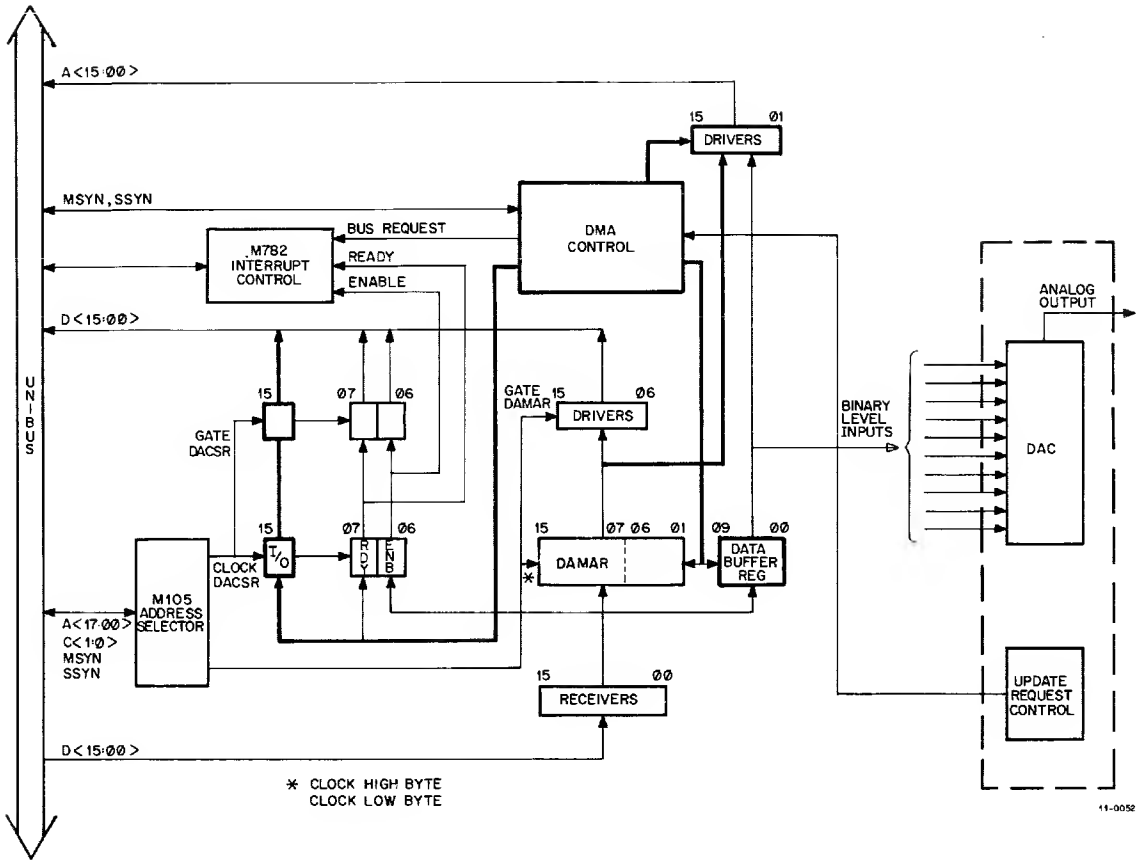


Figure 4-13. DAC-DMA Interface – Block Diagram

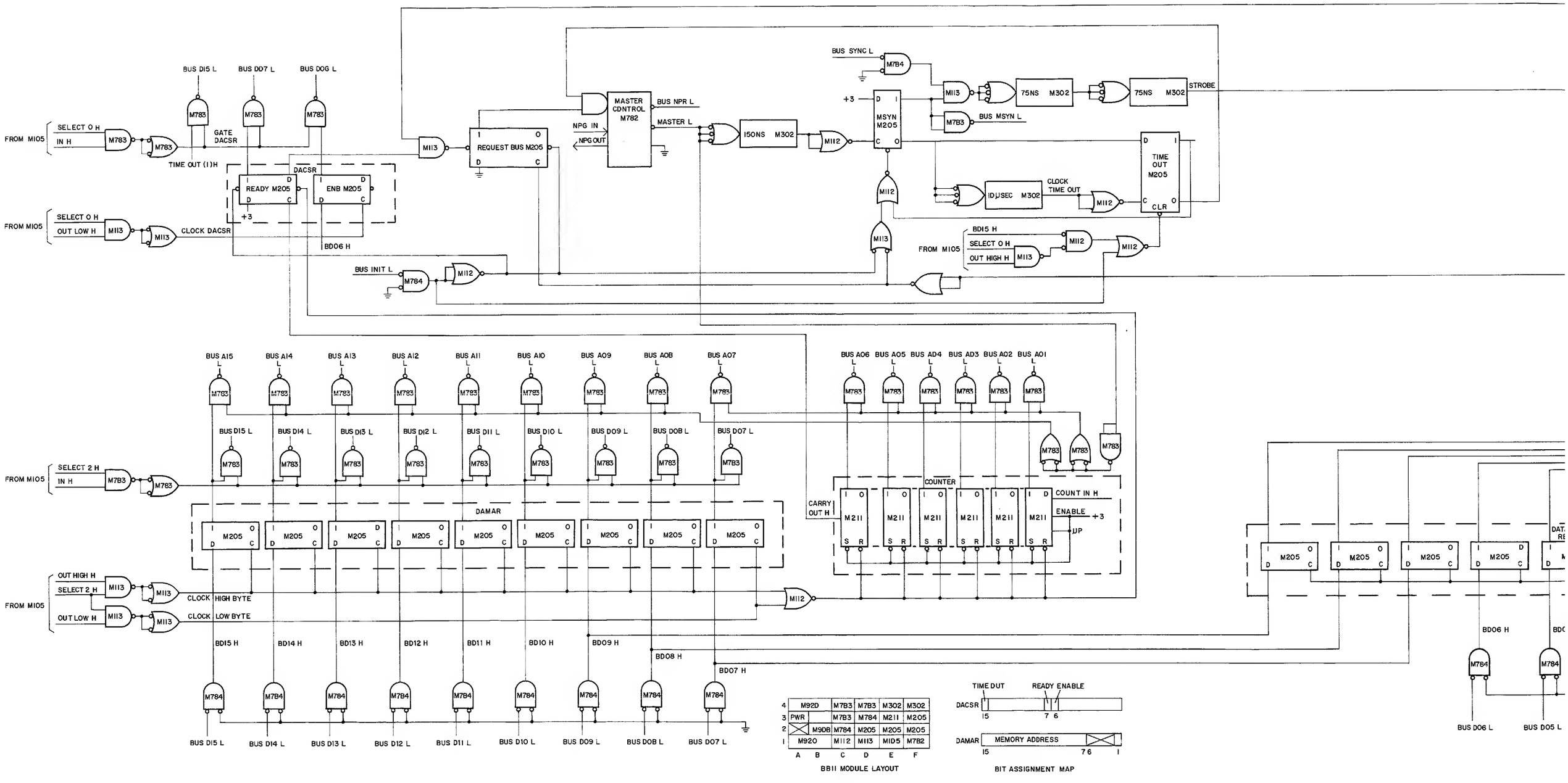
a different function in this interface: the register is used as a memory address register and functions as a combination counter and buffer register.

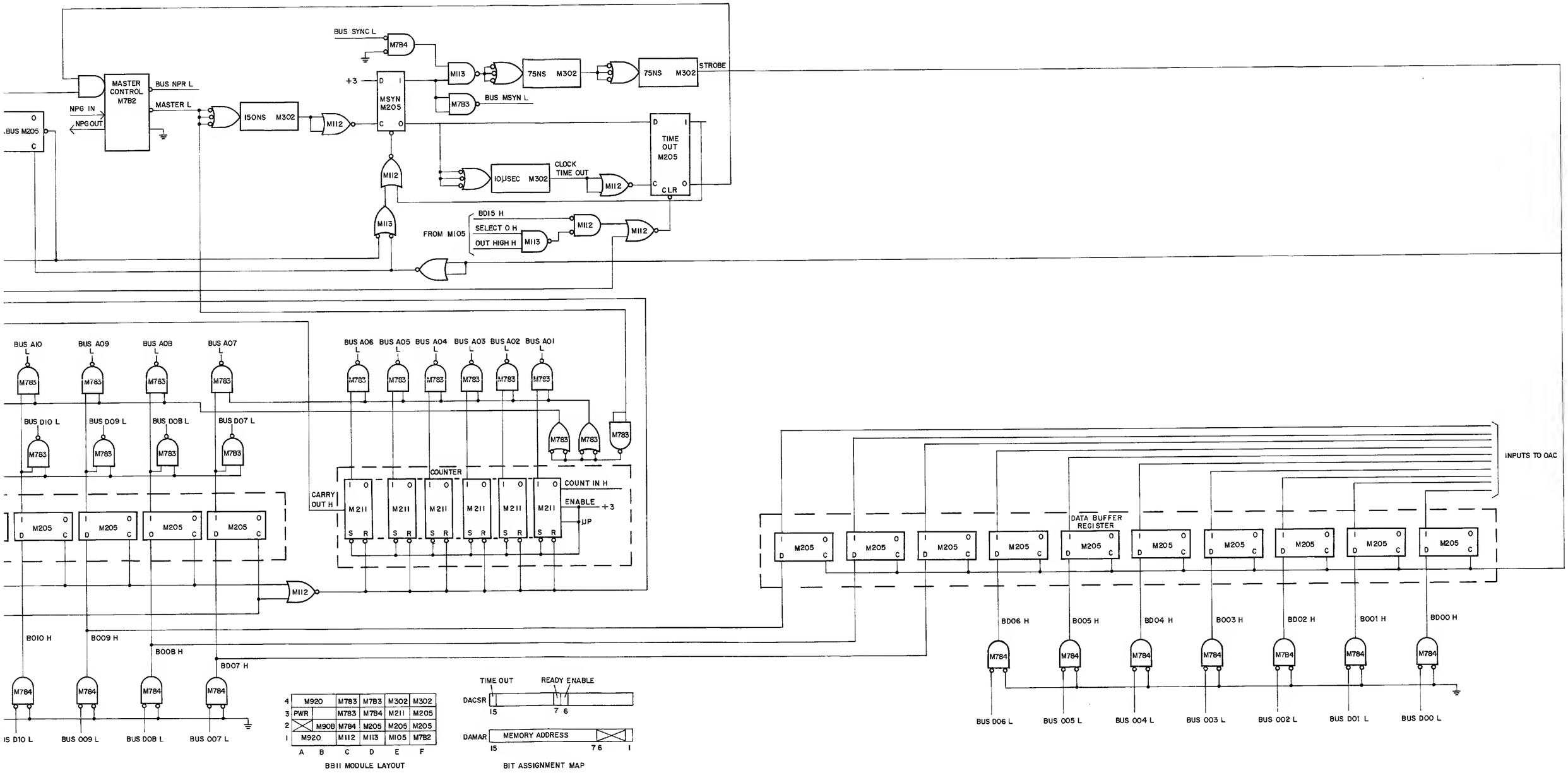
4.7.2 Interface Operation

The interface begins an operation cycle when the memory address register (DAMAR) is loaded from the Unibus. This loading operation clears the 6-bit counter portion of the DAMAR and clears the ready bit (bit 7 of the DACSR). When the DAC generates an update request pulse, the interface requests control of the bus. When the interface gains bus control, it transfers the data word in the location addressed by the contents of the DAMAR into the buffer register. After the transfer is complete, the interface increments the counter and waits for the next update request pulse. If the counter overflows, the ready bit is set and further operation of the interface is inhibited.

4.7.3 Interface Implementation

Figure 4-14 illustrates a possible method of implementing the DAC-DMA interface. The counter in the DAMAR is an M211 Binary Up/Down Counter Module and the time delays are implemented with two M302 Dual Delay Multivibrator Modules. Although the registers and bus gating circuits are similar to the interface described in Paragraph 4.5, control circuits differ because this interface transfers data in the opposite direction.


BUS DI5 L
BUS DD7 L
BUS DOG L
FROM MIO5
SELECT O H
IN H
M783
GATE DACSR
TIME OUT (1)H
DACSR
READY M205
ENB M205
+3
FROM MIO5
SELECT O H
OUT LOW H
M113
CLOCK DACSR
BD06 H
BUS INIT L
M784
M112
M113
REQUEST BUS M205
MASTER CONTROL M782
BUS NPR L
MASTER L
NPG IN
NPG OUT
150NS M302
BUS SYNC L
M784
+3
MSYN M205
M113
75NS M302
75NS M302
STROBE
M783
BUS MSYN L
TIME OUT M205
CLR
1DµSEC M302
CLOCK TIME OUT
BD15 H
SELECT O H
OUT HIGH H
FROM MIO5
BUS AI15 L
BUS AI14 L
BUS AI13 L
BUS AI12 L
BUS AI11 L
BUS AI0 L
BUS AO9 L
BUS AO8 L
BUS AO7 L
BUS AO6 L
BUS AO5 L
BUS AD4 L
BUS AD3 L
BUS AO2 L
BUS AO1 L
BUS DI15 L
BUS DI14 L
BUS DI13 L
BUS DI2 L
BUS DI1 L
BUS DI0 L
BUS DO9 L
BUS DO8 L
BUS DO7 L
FROM MIO5
SELECT 2 H
IN H
M7B3
DAMAR
M205
OUT HIGH H
SELECT 2 H
OUT LOW H
M113
CLOCK HIGH BYTE
CLOCK LOW BYTE
BD15 H
BD14 H
BD13 H
BD12 H
BD11 H
BD10 H
BD09 H
BD08 H
BD07 H
M784
BUS DI5 L
BUS DI4 L
BUS DI3 L
BUS DI2 L
BUS DI1 L
BUS DI0 L
BUS DO9 L
BUS DO8 L
BUS DO7 L
CARRY OUT H
COUNTER
M211
COUNT IN H
ENABLE
+3
UP
BD06 H
BUS DO6 L
BUS DO5 L
4 M92D M7B3 M7B3 M302 M302
3 PWR M7B3 M784 M211 M205
2 M9DB M784 M205 M205 M205
1 M920 M112 M113 MID5 M7B2
A B C D E F
BB11 MODULE LAYOUT
TIME OUT
READY ENABLE
DACSR
15
7 6
DAMAR
MEMORY ADDRESS
15
7 6 1
BIT ASSIGNMENT MAP



	A	B	C	D	E	F
4	M920		M783	M7B3	M302	M302
3	PWR		M783	M784	M211	M205
2		M908	M784	M205	M205	M205
1	M920		M112	M113	M105	M782

Figure 4-14. DAC-DMA Interface – Circuit Schematic

Interface operation begins when the UPDATE REQUEST pulse from the DAC, gated by READY (0) H, sets the request bus flip-flop. If no time-out errors occur, REQUEST BUS (1) H initiates a bus request. When the processor responds, MASTER L gates the DAMAR output to the bus address lines and MSYN is asserted after 150 ns. Then the 10-μs time-out period begins.

The bus C lines are clear; therefore, the addressed location responds to a DATI, placing information on the bus D lines, and asserting SSYN. This signal clears MSYN, clears the bus request flip-flop, and clocks the information from the bus data lines into the data buffer register. After 75 ns, the interface allows the bus address lines to clear, and the interface clocks the counter.

The entire operating cycle is repeated when the next update request pulse is received. Two conditions, however, inhibit this repetition. One condition is a counter overflow that sets the ready bit. The other condition is a time-out error that indicates the interface addressed a nonexistent address and no SSYN signal was returned. If this error occurs, the time-out flip-flop is set, the bus operation is terminated, and an error bit is set in the DACSR. This error condition can be cleared by addressing the DACSR from the bus to load a zero into bit 15 of the register. Figure 4-15 illustrates the timing relationships during interface operation.

Data flow in this interface is as follows. The interface places an address on the bus address lines; data is transferred from the bus to a data buffer register in the device; and, when the transfer is complete, an interface operation places an interrupt vector on the bus data lines.

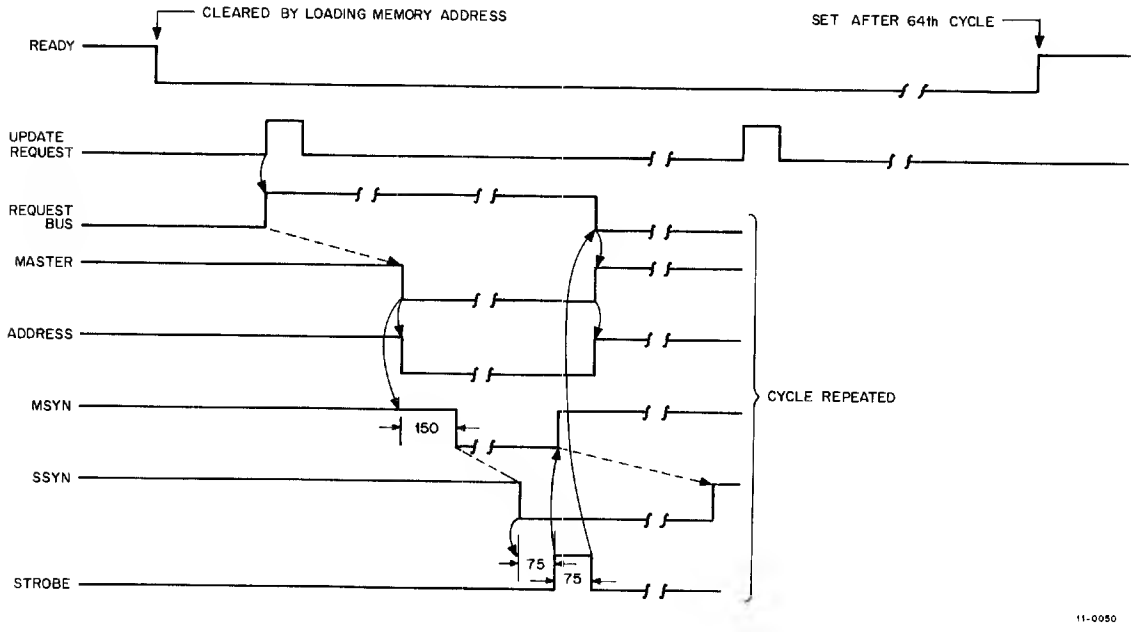


Figure 4-15. DAC-DMA Interface – Timing Diagram

4.8 PDP-11 TO DATA CHANNEL INTERFACE

This paragraph presents a block diagram level description of an interface between the PDP-11 Unibus and a data channel of another computer. Asynchronous timing permits the PDP-11 to operate successfully with the timing of other systems through the implementation of a proper timing interface. Actual design depends on the characteristics of the other computer; therefore, this example is only a representative design.

4.8.1 Data Channel Description

Information on the data channel of the secondary computer includes an address, control and timing signals, and the data being transferred. The address is transmitted to a memory address (MA) register and is stored when the data channel begins operation. Data is supplied to a memory buffer (MB) register on one set of lines and is transmitted from the MB to the interface on a second set of lines. (These may be the same set of lines if the data channel uses bidirectional data lines.) Control and timing signals include: a break request (BRK REQ) signal; a data direction (DATA DIR) signal, which is high (+3V) for a transfer to the channel and low (0V) for a transfer from the channel; a break state (BRK STATE) signal, which is low when the secondary computer is in the break state; and three timing signals transmitted by the channel control to indicate break and address accepted (BRK ACPT), data accepted (DATA ACPT), and data available (DATA AVAIL).

4.8.2 Interface Description

The interface in Figure 4-16 operates as a cycle stealing device on a data channel of another computer and responds as a slave device on the PDP-11 Unibus. The primary purpose of this interface is to transfer data between two processors. Therefore, no provision is made to directly control operation of the secondary computer.

The interface structure includes the data paths between the two processors; the address paths from the Unibus to the secondary processor; a block (BLK) register, which modifies the address transmitted on the data channel in order for it to access one of eight 4K word blocks; and the interface control logic, which determines the timing and gating of the data and control signal flow.

When the PDP-11 addresses a bus location in the seventh block of 4K words (A⟨17:13⟩ = 00110), the interface generates a BRK REQ signal. The address used by the data channel is a 15-bit number assembled by transmitting the information on lines A⟨12:01⟩ as the 12 least significant bits, and transmitting the contents of the BLK register as the 3 most significant bits. The interface responds to 4K word (8K byte) addresses on the Unibus, but can access 32K words in the data channel. Only 4K words are accessible for any setting of the BLK register.

4.8.3 Interface Operation

When performing transfers with the data channel computer, the PDP-11 first loads the desired block number into the BLK register, which is addressed as a device register (A⟨17:13⟩ = 11111). The PDP-11 then addresses a location in the 4K block of Unibus addresses which have A⟨17:13⟩ = 00110. The corresponding location in the 4K block selected by the BLK register is selected by the data channel. A data transfer is accomplished by issuing a BRK REQ signal, an address, and setting the DATA DIR level; receiving a BRK ACPT pulse; gating data through the interface; and receiving either a DATA ACPT or a DATA AVAIL pulse to generate SSYN.

4.8.4 Interface Implementation

Figure 4-17 illustrates details of a possible control circuit for implementing the PDP-11 to data channel interface. The BRK REQ flip-flop is set when the bus address is in the specified 4K block and cleared by the BRK ACPT H pulse from the data channel. The SSYNF flip-flop is set by either a DATA ACPT H or a DATA AVAIL H pulse,

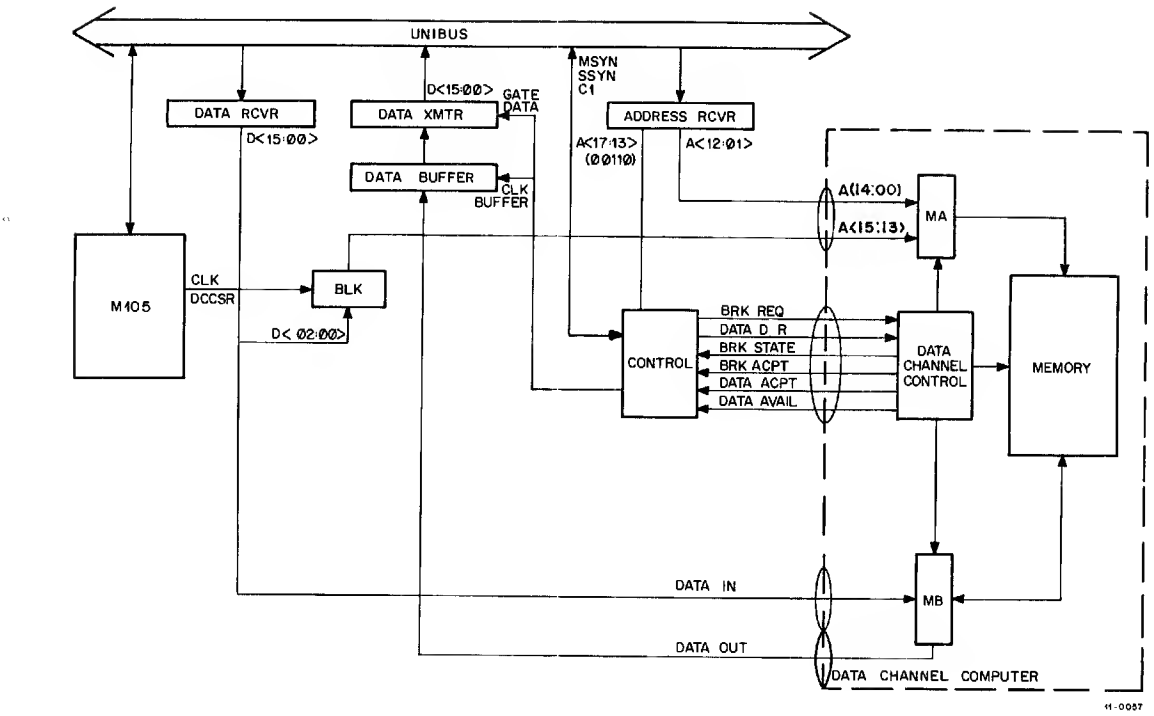


Figure 4-16. PDP-11-to-Data Channel Interface – Block Diagram

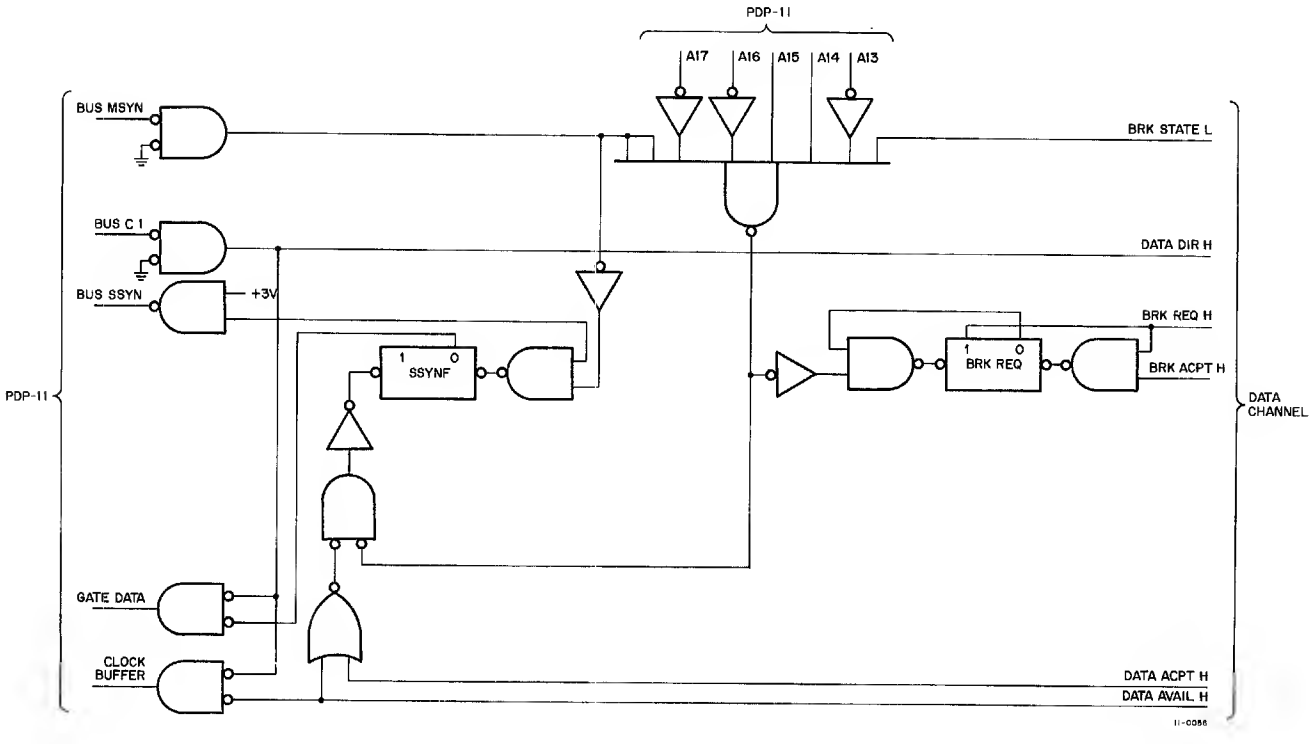


Figure 4-17. PDP-11-to-Data Channel Interface – Control Circuit Schematic

depending on the direction of the data transfer. The SSYNF flip-flop is cleared when MSYN is cleared. The DATA AVAIL H pulse also clocks data into a buffer register, which is gated to the bus when the SSYNF flip-flop is set. This provides a data transfer to the bus. This implementation assumes that the data channel computer operates with logic levels similar to the PDP-11. Thus, a high (+3V) level is a logic 1 and a low (0V) level is a logic 0.

4.9 PDP-11 TO PDP-11 INTERFACE

The PDP-11 to PDP-11 interface permits devices on one PDP-11 Unibus to address locations on a second PDP-11 Unibus. Two possible applications of this interface are:

a. Connecting two PDP-11 processors so that one processor executes its own programs and also controls execution of programs by a second slave processor.

b. Monitoring operation of a Unibus system by a processor that is not in the same system.

4.9.1 Interface Description and Operation

The PDP-11 to PDP-11 interface (Figure 4-18) consists of one M782 Interrupt Control, one M105 Address Selector Module, several sets of bus receivers and bus drivers, and some control circuits. The M105 Module is connected to the primary Unibus and the M782 Module is connected to the secondary Unibus. The major flow of data within the interface is from one Unibus, through bus receivers and gated bus drivers, to the other Unibus. Data also flows from the primary Unibus to the control circuit.

When the primary Unibus requests interface operation, the interface requests control of the secondary Unibus, and the primary Unibus is connected to allow data transfers with the secondary Unibus. The interface is designed to recognize addresses in the seventh 8K (byte) field of primary Unibus addresses, and convert them to addresses in any selected 8K (byte) field of addresses on the secondary Unibus. This is equivalent to replacing one 8K field of bus addresses on the primary Unibus with an 8K field of addresses from the secondary Unibus. The field on the secondary Unibus is selected by loading a field select register in bits 5 through 1 of the Unibus to Unibus control status register (UUCSR) in the interface. The contents of this 5-bit register are used for the five most significant bits of the secondary bus address. This may be loaded by a MOV BLOCK, UUCSR instruction.

The interface can be operated in one of two modes. In the first mode, a single bus operation may access the secondary Unibus by addressing the seventh 4K field on the primary Unibus. The interface recognizes the address, requests control of the secondary bus, and interconnects the two sets of bus lines when it receives control. The interface releases control of the secondary Unibus when the transfer is complete.

In the second mode of operation, if the hold bit is set in the UUCSR, the interface requests control of the secondary bus and maintains bus mastership until the hold bit is cleared. This mode of operation permits the primary bus to conduct a series of data transfers with the secondary bus at the full bus transfer speed. The only additional time required for a transfer is the signal transmission time within the interface. The time delays caused by waiting for the secondary bus to grant bus mastership to the interface are eliminated; therefore, a disk on the primary bus can transfer data to locations on the secondary bus with the lowest possible latency delays.

4.9.2 Interface Implementation

In Figure 4-18, Unibus 1 (the primary Unibus) controls the interface and conducts data transfers through the interface to Unibus 2 (the secondary Unibus). The interface control circuit is shown in greater detail in Figure 4-19. This figure illustrates the UUCSR device register, an address recognition circuit, a MSYN delay circuit, and several gating circuits.

When the address asserted on the primary Unibus is in the seventh 8K field (A⟨17:13⟩ = 00110), the STEAL signal is asserted in the interface. The STEAL L signal causes an NPR request on the secondary unibus. (If the

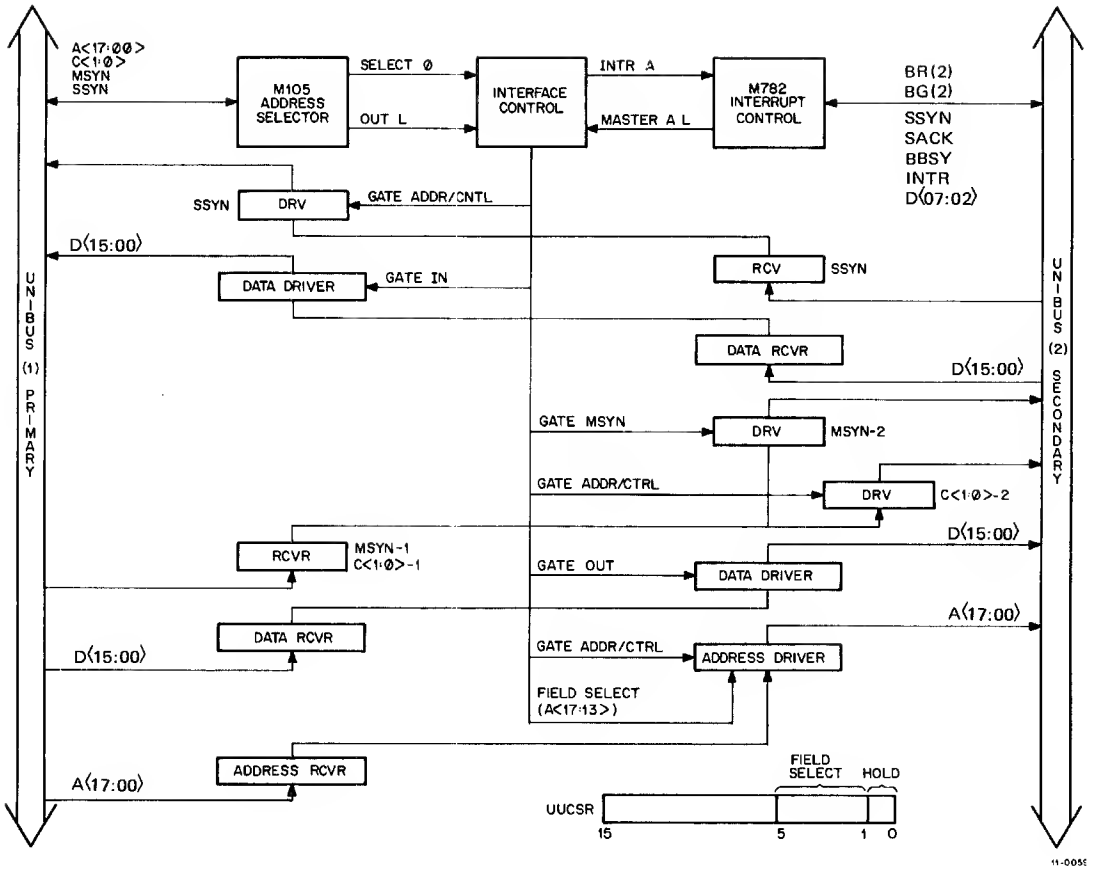


Figure 4-18. PDP-11-to-PDP-11 Interface – Block Diagram

hold flip-flop is set, the interface is already bus master and no further bus requests are initiated.) When the interface becomes bus master, the MASTER A L signal is asserted. The combination of STEAL L and MASTER A L produces gating signals to interconnect the two sets of bus lines as shown in Table 4-1. The MSYN delay circuit regenerates the 150-ns deskewing period between the assertion of the A and C lines and the assertion of MSYN on the secondary Unibus. This prevents delays in signal transmission within the interface from affecting the timing on either Unibus.

Table 4-1
Bus Line Gating

Signal	From	To	Gated By	Remarks
A(17:13)	Block Register	Unibus 2	GATE ADDR/CTRL	
A(12:00)	Unibus 1	Unibus 2	GATE ADDR/CTRL	
C(1:0)	Unibus 1	Unibus 2	GATE ADDR/CTRL	
D(15:00)	Unibus 2	Unibus 1	GATE IN	C1=0 (DATI or DATIP)
D(15:00)	Unibus 1	Unibus 2	GATE OUT	C1=1 (DATO or DATOB)
MSYN	Unibus 1	Unibus 2	GATE MSYN	
SSYN	Unibus 2	Unibus 1	GATE ADDR/CTRL	

4.9.3 Interface Programming

To type a character on the secondary Unibus system Teletype, the processor on the primary Unibus executes the following sequence of instructions:

```
MOV     #76, UUCSR
MOV     CHAR, XPB
```

where: UUCSR is the control register in the interface
CHAR is the location containing the character
XPB has the value 157566

The address transmitted on the secondary Unibus is the 13 least significant bits of XPB (17566) with the contents of the field select register appended as the 5 most significant bits. The address is therefore 777566, which is the address of the system Teletype TPB.

A block of data is transferred to locations on the secondary Unibus by setting the hold bit when the field register is set:

```
MOV     FIELD, UUCSR
```

where: FIELD is the number of the 4K field of word addresses to be accessed on the secondary Unibus (with a 1 appended to set the hold bit)

Data transfers then access locations in that address field when the address on the primary Unibus has A⟨17:13⟩ equal to 00110 and A⟨12:00⟩ equal to the desired value of A⟨12:00⟩ on the secondary Unibus. If FIELD has the value 11_8, the address 140020_8 is translated to the address 100020_8.

4.10 MEMORY INCREMENT INTERFACE

A Memory Increment interface is used to increment a device-selected memory location without requiring processor intervention. The interface operates by performing a DATIP, DATO sequence with the selected memory location. This sequence is the same as processor operation with a destination operand. Data read from the memory location by the DATIP transfer is loaded into a counter register. A count cycle is initiated, and the count is allowed to ripple through the register. The new value is written back into the memory during the DATO cycle.

A more flexible implementation uses an adder instead of a counter. The contents of the memory location are brought into a latch, which forms one input to the adder. The other input can be under device or program control, or a simple increment operation can be executed by providing a carry into the least significant bit. The adder permits arbitrary values to be added to the selected location. This expanded interface is not described in subsequent discussions.

This interface differs from previous interface examples because the device output is used as a bus address rather than as data. The operations performed on the data in these locations effectively generate a time-interval histogram directly in core memory.

4.10.1 Interface Description and Operation

In Figure 4-20, the analog-to-digital converter (ADC), which is an external device that supplies the digital value used as an address, is not part of the interface. One device register is used in the interface. The device register contains the enable bit which controls access of the interface to the Unibus, and contains the five field select bits used to select a 4K word block of bus addresses that the interface uses for conducting transfers. The counter and transmitters are under interface control. The interface provides all timing and control signals for one cycle of operation and each successive cycle of operation is initiated by a signal from the device. In this example, the device is an ADC similar to the ADC described in Paragraph 4.4. The signal that starts a cycle of interface

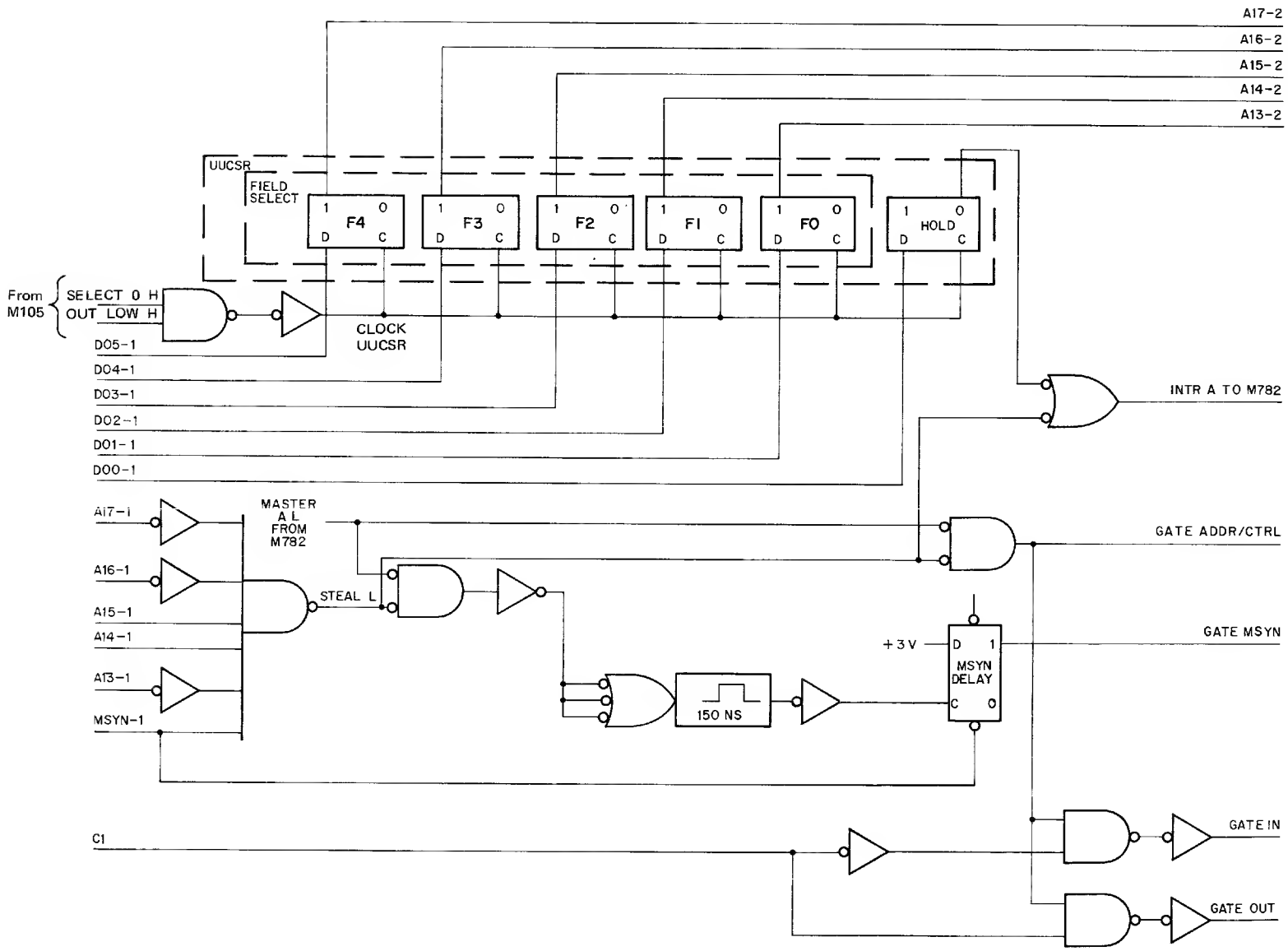


Figure 4-19. PDP-11-to-PDP-11 Interface — Control Circuit Schematic

operation is the CONVERSION COMPLETE signal. The time between cycles of operation is under ADC control. The ADC includes a circuit that determines the time between generation of a start conversion signal (to complete the next conversion) and generation of a conversion complete signal.

When an interface operating cycle begins, the interface requests control of the Unibus by asserting an NPR signal. When it becomes bus master, the interface asserts a bus address, which is assembled by transmitting the digital output of the ADC as the 12 least significant bits (A<12:01>) and the contents of the field select register as the 5 most significant bits (A<17:13>). The interface asserts C0 (the Unibus command for a DATIP), and asserts MSYN. The memory responds with SSYN and a word of data. The SSYN signal causes the interface to load the counter and clear MSYN. After 75 ns, the interface clears C0, asserts C1, and increments the counter. The outputs of the counter are gated to the bus data lines when the interface asserts C1 for the DATO sequence. The MSYN signal is asserted when the count has had time to ripple through all stages of the counter. The address and control lines remain asserted until the end of the second bus transfer (which is a DATO because C1 is asserted). When the memory asserts SSYN for the second time, the interface drops MSYN, waits 75 ns, and drops the A, C, and D lines. The ready bit is cleared in preparation for the beginning of the next cycle.

4.10.2 Interface Implementation

The control logic (see Figure 4-21) for the interface includes: a timing circuit, which includes 2 one-shots similar to those supplied on the M302 Dual Delay Multivibrator Module; three pulse amplifiers similar to those supplied on the M606 Pulse Generator Module; and a delay line similar to those available on the M310 Delay Line Module. The relationships of various timing signals are shown in Figure 4-22.

The C flip-flop asserts either the C0 or C1 line to generate either a DATIP or DATO operation. The MSYN flip-flop is set and cleared twice during a cycle of device operation. The start conversion signal is asserted by the END L signal and also by the clocking signal (LOAD MICSR), which loads the interface enable bit. Clocking signal assertion prevents an unwanted cycle of data transfers that might occur if the enable bit is set without clearing the ready bit and the device has CONVERSION COMPLETE asserted.

4.10.3 Interface Use

The processor initiates interface operation by executing the following instruction:

```
MOV          INIT, MICSR
```

where: MICSR is the device register containing the enable bit and the field select bits

INIT is the octal number of the desired block of addresses shifted left one bit, with bit 0 set to 1 to enable the interface

The interface operates until the enable bit is cleared by the processor. This may be accomplished by an interrupt service routine responding to a real-time clock, another device, or operator intervention.

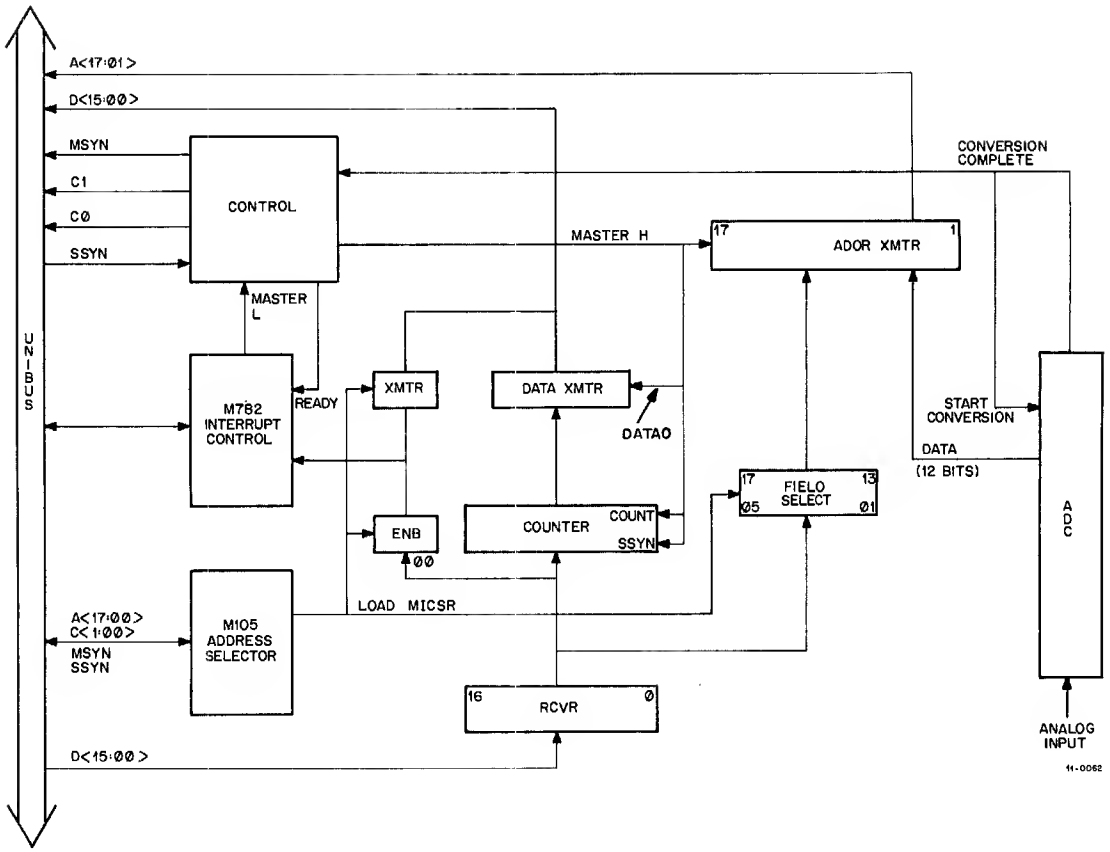


Figure 4-20. Add-to-Memory Interface – Block Diagram

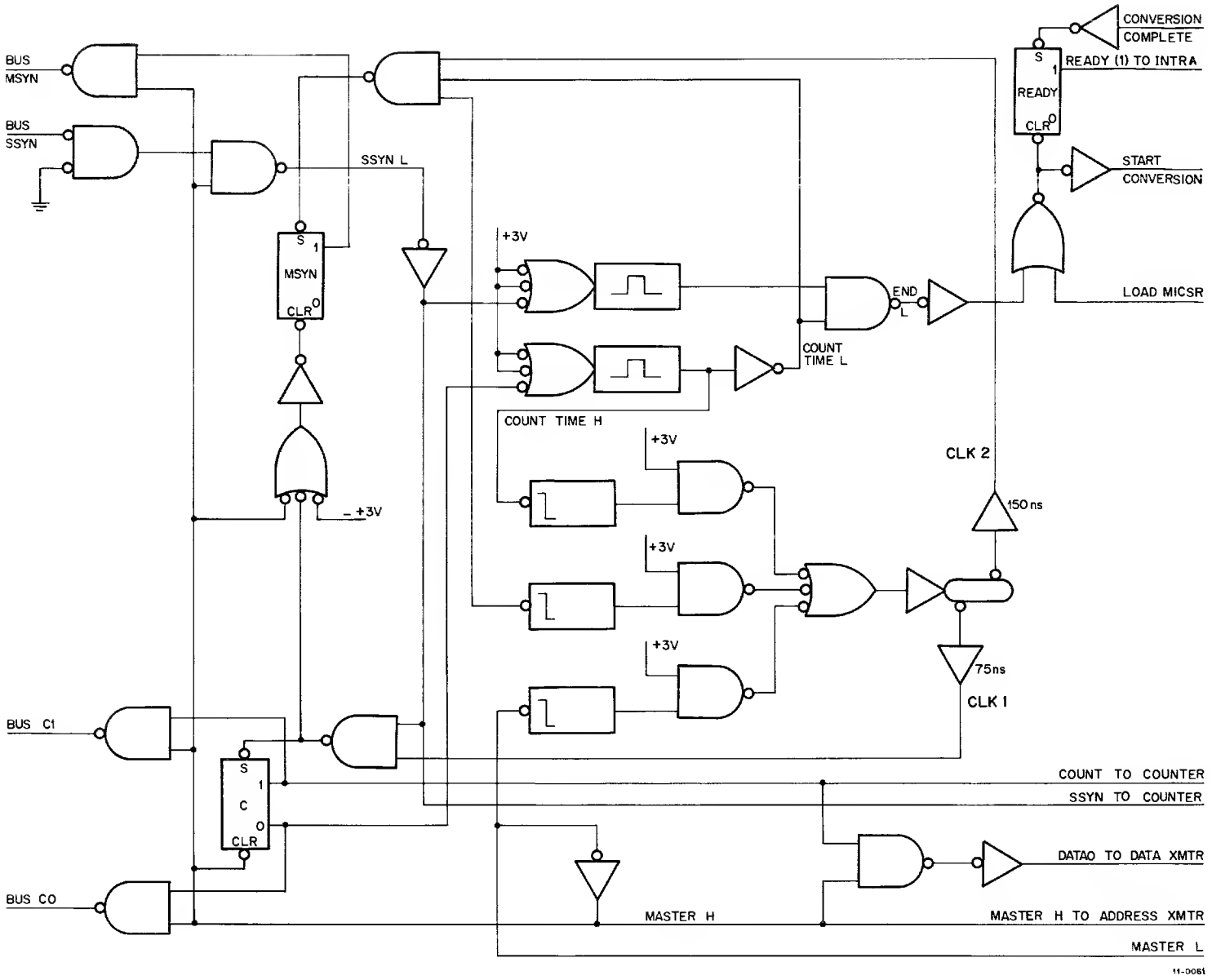


Figure 4-21. Add-to-Memory Interface – Control Circuit Schematic

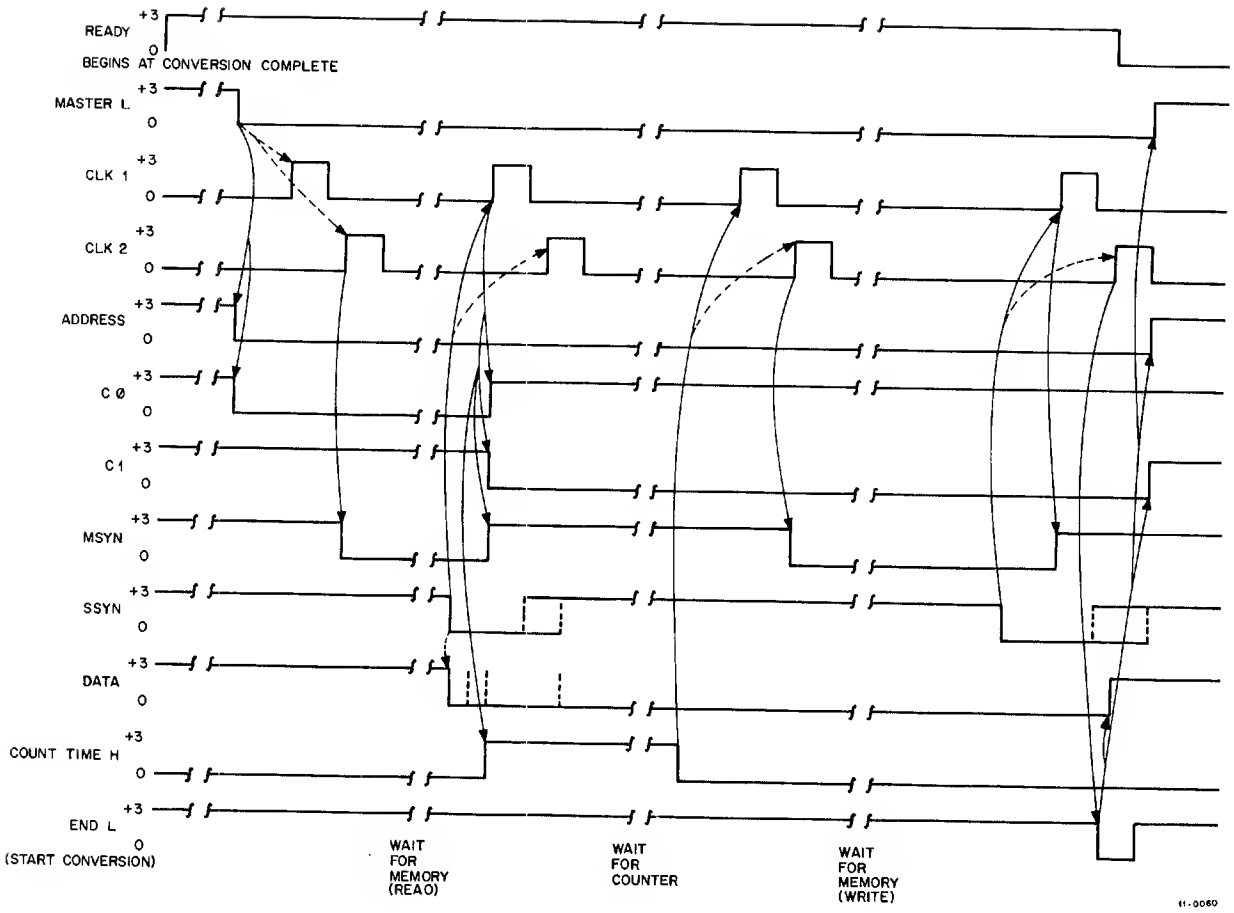


Figure 4-22. Add-To-Memory Interface – Timing Diagram

Appendix A
Processor Block Diagram

A.1 INTRODUCTION

This description of the KA11 Processor is intended to give the reader sufficient knowledge of the processor to understand the processor's role in the control of bus activity, and the use of bus operations involved in normal processing. It is beyond the scope of this Appendix to provide a detailed description of the processor. A detailed description of the processor, including theory of operation and logic design, is provided in Volume 1 of the KA11 Processor Manual. This Appendix describes the block diagram, and then briefly discusses major states and bus cycles.

A.2 BLOCK DIAGRAM

Figure A-1 is a simplified block diagram of the processor and is divided into four major functional areas: Unibus interface, data paths, general-purpose registers, and timing state and control logic. Each of these areas is discussed separately in subsequent paragraphs to bring out the logical structure of the processor.

A.2.1 Unibus Interface

The processor is connected to the bus data and address lines through input buffering and output driving gates. The bus control and control transfer lines are connected to the bus through the processor timing state and control logic. Most of the data originates, and ultimately is stored at or received by, locations external to the processor. Therefore, during normal processing the majority of information transfers take place through the bus interface.

A.2.2 Data Paths

The data paths perform all modification and routing of data within the processor. In effect, this is where normal processing and computation occurs. The data paths consist of A and B input gating, an adder, and output gating.

A.2.3 General-Purpose Registers

In general, data storage is external to the processor, and these storage locations are accessed through the Unibus. The only internal storage locations available for general data, are the flip-flop storage elements (general-purpose registers). These registers are used as index registers for address calculations, or as storage for frequently used data. Six of these registers are used by hardware functions, one is used as a program counter, and one is used as a stack pointer.

All information transfers are through the data paths. The bus interface and general-purpose registers interconnect only through the data paths, so that frequent transfers of data set up a flow of information in a figure-eight pattern (see Figure A-2). This flow is from the Unibus to the bottom of the data paths, from the top of the data paths to the general-purpose registers, from the general-purpose registers to the bottom of the data paths, and from the top of the data paths to the Unibus.

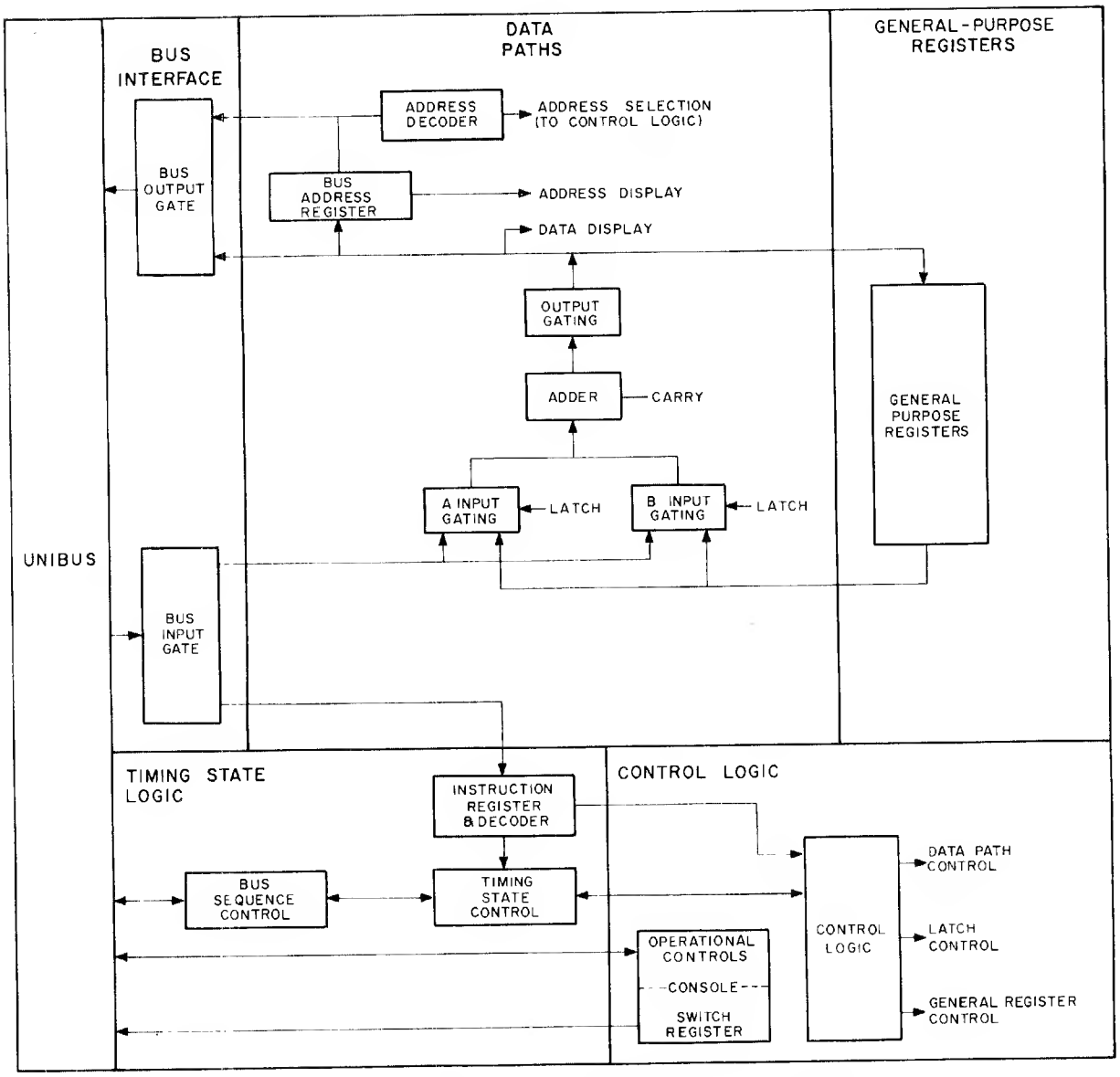


Figure A-1. PDP-11 Processor Block Diagram

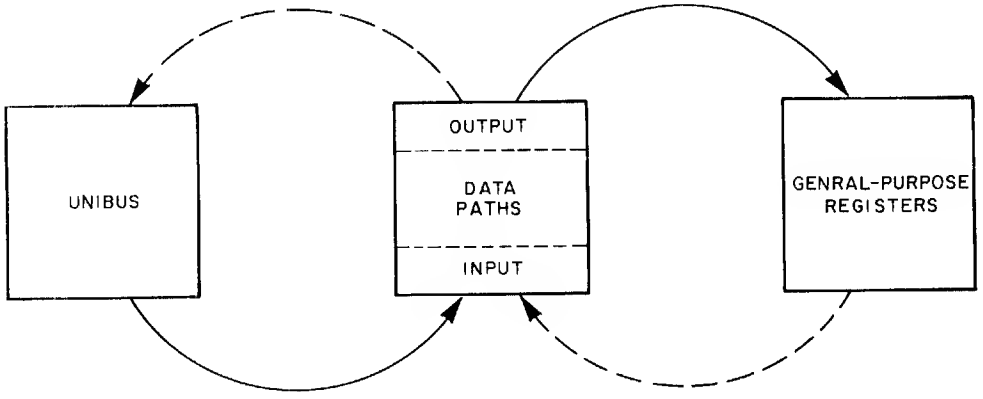


Figure A-2. Data Flow

A.3 MAJOR STATES

The KA11 Processor has five major states: fetch, source, destination, execute, and service. The first four states are used during normal processor operation, while the last state (service) is used during special operations, such as traps and interrupts. A brief description of major states and related functions is presented in Table A-1.

Table A-1
Processor Major States

Major State	Operand	Function
Fetch	All instructions	Locates and decodes instruction.
Source	Double-operand group only	Decodes source field and uses data paths to transfer addressed data to proper location.
Destination	Double-operand group	Decodes destination field and uses data paths to transfer addressed data to proper locations. Must follow source major state in this instance.
	Any other group	Decodes destination field of instruction and transfers data to appropriate location. Other groups with destinations are: all single-operand, rotate/shift, and subroutine call. Also, jump and swap byte instructions of the operate group.
Execute	Most instructions	Performs operation specified by the instruction (examples: arithmetic operations, tests, etc.).
Service	Any instruction	Performs special operation such as interrupts and traps.

The fetch major state locates and decodes an instruction. When this major state is completed, the processor enters another major state, depending on the type of instruction decoded. It is possible to go from fetch to any other state, including back to fetch. Every instruction must first enter the fetch major state.

The source major state decodes the source field of a double-operand group instruction and transfers data to appropriate locations. The source major state is entered only if the instruction belongs to the double-operand group.

The destination major state decodes the destination field of the appropriate instruction. Destination fields are present in all double-operand, single-operand, rotate/shift, and subroutine call instructions.

The execute major state uses the information from previous major states to perform the specified operation. During this state arithmetic operations, logic functions, and tests are performed.

The service major state is used to execute special operations, such as interrupts, traps, etc.

Some typical major state sequences are given in Table A-2.

Table A-2
Typical Major State Sequences

	Major States				
Instruction	Fetch	Source	Dest.	Execute	Service
DOUBLE-OPERAND GROUP (single-word format)					
1. Both source and destination address modes 1, 2, or 3	X	X	X	X	
2. Both source and destination address modes are 0	X			X	
3. Only source address mode is 0	X		X	X	
4. Only destination address mode is 0	X	X		X	
DOUBLE-OPERAND GROUP (2- or 3-word formats)	X	X	X	X	
SINGLE-OPERAND GROUP	X		X	X	
TRAP GROUP	X				X
BRANCH GROUP (condition met)	X			X	
BRANCH GROUP (condition not met)	X				

A.4 BUS CYCLES

Whenever data is transferred between devices, either a DATI (data transfer in) or DATO (data transfer out) bus cycle is used. The direction of data transfers is relative to the master (controlling) device.

In addition to the basic DATI and DATO bus cycles, there is also a modified DATI, known as DATIP (data transfer in, pause) and a modified DATO, known as DATOB (data transfer out, byte).

During the fetch major state, the processor always performs one DATI bus operation. For the source and destination major states, when applicable, the processor may do from one to three DATI operations in each major state, depending on the address mode and the number of levels of indirect addressing. If no indirect addressing is used, the major state is skipped for that word of data.

The execute and service major states vary for different instructions. The states are shown, along with the related instructions, in Appendix B.

Appendix B
Instruction Set

B.1 INTRODUCTION

This Appendix provides a description of the major states and bus cycles used by each instruction in the PDP-11 Instruction Set. The discussion is divided into two parts: a brief description of the six groups of instructions, and a discussion of bus cycles used by each instruction.

A detailed description of processor major states, bus cycles, and instruction decoding is contained in the KA11 Processor Manual. A detailed description of the instruction set is given in the PDP-11 Handbook.

B.2 INSTRUCTION GROUPS

The PDP-11 Instruction Set is divided into six major groups. A brief description of each group is given below.

Specific instructions within each group are covered in Paragraph B.3.

a. Double-Operand Group – These instructions permit certain functions to be performed on both the source and destination data. Typical functions include: move, compare, add, and subtract. Instructions in this group can specify either byte or word operands except add and subtract.

b. Single-Operand Group –These single-operand instructions permit certain functions to be performed on the destination data. Typical functions include: clear, complement, increment, decrement, etc. The operand may be either a byte or a full word.

c. Conditional Branches – These test instructions check the results of an operation and cause the program to branch according to test results. Typical tests include: equal to zero, overflow, carry, etc.

d. Operate Group – These direct instructions, such as HALT and WAIT, also contain trap instructions, and the Return From Interrupt (RTI) instruction.

e. Subroutine Group – This group contains the two instructions for using subroutines: Jump To Subroutine (JSR) and Return From Subroutine (RTS).

f. Condition Codes Group – These instructions set condition code bits in the status hardware register to define the processor status (PS) word. The PS indicates whether or not certain conditions (such as zero, carry, etc.) are present.

B.3 MAJOR STATES

A general description of processor major states is given in Appendix A. The prime purpose of Appendix B is to cover specific instructions and list the bus cycles that occur as the instruction enters each major state.

B.3.1 Fetch

All instructions must enter the fetch major state. A DATI is performed to bring the instruction into the processor. The instruction is decoded, and the processor enters any one of the other four major states, depending on the instruction. If it is a double-operand instruction, it can enter the source major state. If it is a single-operand, rotate/shift, Jump (JMP), Jump To Subroutine (JSR), or Branch (BR), it can enter the destination major state. If it is a branch with the condition met, internal address, RTI, or RTS, it enters the execute major state. If it is an interrupt or trap, it enters the service major state.

B.3.2 Source and Destination

Both the source and destination major states are used for address calculations. When address mode 0 is used in either of these states, the state is skipped because address calculations is not necessary. Bus cycles for the remaining 7 address modes are listed in Table B-1.

Table B-1
Address Modes
(Source or Destination)

Address Modes	Bus Cycles
1, 2, 4	DATIP
3, 5, 6	DATI DATIP*
7	DATI DATI DATIP*

*DATI if in SOURCE or DEST major state and instruction is CMP, BIT, TST, or JMP.

B.3.3 Execute

Since most of the instruction processing takes place during execute, it is the most complex major state. Six tables (B-2 through B-7) provide coverage of bus cycle information for each instruction. Each table represents a specific group of instructions, and lists all instructions within that group. The tables list the major states each instruction enters, and the bus cycles that occur during execute. These tables are:

Table B-2	Double-Operand Group
Table B-3	Single-Operand Group
Table B-4	Operate Group
Table B-5	Subroutine Group
Table B-6	Condition Code Operators
Table B-7	Conditional Branches

Table B-2
Double-Operand Group

Instruction	Major States F S D E	Bus Cycles in Execute
MOV(B)	† * * †	DATO(B)††
CMP(B)	† * * †	NONE
BIT(B)	† * * †	NONE
BIC(B)	† * * †	DATO(B)††
BIS(B)	† * * †	DATO(B)††
ADD	† * * †	DATO††
SUB	† * * †	DATO††

†always used. †† modified DATO or DATOB (following DATIP).
*may or may not be used.

NOTE
Whenever destination address mode is 0, no DATO(B) bus cycle is performed.

Table B-3
Single-Operand Group

Instruction	Major States F S D E	Bus Cycles in Execute	Remarks
CLR(B)	† 0 * †	DATO(B)††	
COM(B)	† 0 * †	DATO(B)††	
INC(B)	† 0 * †	DATO(B)††	
DEC(B)	† 0 * †	DATO(B)††	
NEG(B)	† 0 * †	DATO(B)††	
ADC(B)	† 0 * †	DATO(B)††	
SBC(B)	† 0 * †	DATO(B)††	
TST(B)	† 0 * †	None	
ROR(B)	† 0 * †	DATO(B)††	
ROL(B)	† 0 * †	DATO(B)††	
ASR(B)	† 0 * †	DATO(B)††	
ASL(B)	† 0 * †	DATO(B)††	
JMP	† 0 * 0	None	
SWAB	† 0 † †	DATO(B)††	Goes direct to service major state

†always used
*may or may not be used
0 never used
††modified DATO or DATOB (following DATIP).

NOTE

1. Whenever destination address mode is 0, no DATO(B) bus cycle is performed.
2. In the case of the JMP instruction, the last bus operation in destination major state is not performed.

Table B-4
Operate Group

Instruction	Major States F S D E	Bus Cycles in Execute	Remarks
HALT	† 0 0 0	None	Transfers control to console.
WAIT	† 0 0 0	None	Processor stays in service major state until a bus request is received.
RESET	† 0 0 0	None	Transfers control to console.
RTI	† 0 0 †	DATI DATI	
IOT	† 0 0 0	None	See note below.
EMT	† 0 0 0	None	See note below.
TRAP	† 0 0 0	None	See note below.

† always used.
0 never used.

NOTE

For this instruction, bus cycles are performed during the service major state. Bus cycles for these instructions are given in Table B-8.

Table B-5
Subroutine Group

Instruction	Major States F S D E	Bus Cycles in Execute	Remarks
JSR	† 0 † †	DATO	
RTS	† 0 0 †	DATI	

† always used.
0 never used.

NOTE
For the JSR instruction, the last bus operation in the destination major state is not performed.

Table B-6
Condition Code Operators

Instruction	Major States F S D E	Bus Cycles in Execute
CLC	† 0 0 †	None
CLV	† 0 0 †	None
CLZ	† 0 0 †	None
CLN	† 0 0 †	None
SEC	† 0 0 †	None
SEV	† 0 0 †	None
SEZ	† 0 0 †	None
SEN	† 0 0 †	None

†always used.
0 never used.

Table B-7
Conditional Branches

Instruction	Major States F S D E	Bus Cycles in Execute
BR	† 0 0 †	None
BNE	† 0 0 †	None
BEQ	† 0 0 †	None
BGE	† 0 0 †	None
BLT	† 0 0 †	None
BGT	† 0 0 †	None
BLE	† 0 0 †	None
BPL	† 0 0 †	None
BMI	† 0 0 †	None
BHI	† 0 0 †	None
BLOS	† 0 0 †	None
BVC	† 0 0 †	None
BVS	† 0 0 †	None

Table B-7
Conditional Branches (cont)

Instruction	Major States F S D E	Bus Cycles in Execute
BCC or BHIS	† 0 0 †	None
BCS or BLO	† 0 0 †	None

†always used.
0 never used.

B.3.4 Service

In relation to the instruction set, the service major state has two primary functions: to implement certain asynchronous operations, such as console operations, and to process traps and interrupts.

For transfers to console control, such as those caused by HALT or RESET instructions, no bus cycles are performed. During trap and interrupt instructions, however, bus cycles occur during service. These cycles are listed in Table B-8.

Table B-8
Bus Cycles During Service

Instruction	Major States F S D E S	Bus Cycles in Service
IOT	† 0 0 0 †	DATO DATO DATI DATI
EMT	† 0 0 0 †	DATO DATO DATI DATI
TRAP	† 0 0 0 †	DATO DATO DATI DATI

†always used.
0 never used.

Table C-3
BB11 Power Pin Assignments

Pin	Power
A1	-15V
A2	+ 5V
B1	-15V
B2	-15V
C1	-15V
C2	GND
D1	-15V
D2	GND
E1	-15V
E2	GND
F1	-15V
F2	GND
H1	-15V
H2	+ 5V
J1	-15V
J2	+ 5V
K1	-15V
K2	+ 5V
L1	-15V
L2	+ 5V
M1	-15V
M2	+ 5V
N1	GND
N2	-25V
P1	GND
P2	LTC L
R1	GND
R2	ACLO L
S1	GND
S2	DCLO L
T1	GND
T2	+ 8V
U1	GND
U2	+ 8V
V1	GND
V2	+ 8V

NOTE
Power is in module slot A3 of all system units mounted in BA11 mounting boxes equipped with H720 power supplies.

digital equipment corporation